Information System Agnostic Ancestry for Digital Objects

DOCTORAL THESIS

FOR THE DEGREE OF A
DOCTOR IN INFORMATICS

AT THE FACULTY OF ECONOMICS,
BUSINESS ADMINISTRATION AND
INFORMATION TECHNOLOGY
OF THE
UNIVERSITY OF ZURICH

by
Stephan Jakob Benedikt Heuscher

from
Herisau, AR, Switzerland

Accepted on the recommendation of
PROF. DR. A. BERNSTEIN
PROF. DR. M. GUERCIO

2010

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich herewith permits the publication of the aforementioned dissertation without expressing any opinion on the views contained therein.

Zurich, April 14. 2010

The Vice Dean of the Academic Program in Informatics: Prof. Dr. H. C. Gall

Acknowledgements

I would first like to thank my initial advisor Prof. Dr. Klaus R. Dittrich, who was a great inspiration to me and always asked the right questions. Regrettably he did not see this thesis finished.

I am grateful to my second advisor Prof. Dr. Abraham Bernstein for accepting the role as advisor and for seeing this thesis to a happy end.

I would like to thank Prof. Dr. Maria Guercio of the University of Urbino, Italy, for standing in as my co-advisor.

I would also like to thank the Department of Informatics of the University of Zurich as well as the Swiss Federal Archives and in particular Prof. Dr. Christoph Graf for making this work possible. Thanks are also due to the State Archives of Appenzell Ausserrhoden, in particular to Dr. Peter Witschi for allowing me to test my findings in a real-world scenario.

I am particularly indebted to Georg Büchler, Dr. Peter Keller-Marxer, Dr. Steve Casera, Dr. Gaelene Kramers, Heinz Gnehm, Thomas Zürcher, and Prof. Dr. Seamus Ross for their intellectual challenges, support, and careful reviews of this thesis.

I also owe many thanks to my parents Lucia and Jakob Heuscher as well as my grandmother Rosa Heuscher for their support and encouragement during my work on the thesis.

Last but not least, I would like to express my thanks and gratitude to my wife Julia for relieving me of the duties of daily life but mostly for supporting me and having confidence in me.

Abstract

More and more information is becoming available in digital form, most of it derived from digital sources. Digital information is made available as digital objects composed of a sequence of bits and managed by information systems. To date, these digital objects have no independent identification which can be refered outside of a specific information system. However, they normally outlive these systems and can be copied to other systems. In order for the ancestry of digital information to span more than one generation, each source, and therefore each digital object, must provide information on its ancestry. This information needs to be accessible independent of a specific information system.

In this thesis, we present the theoretical foundation for the identification of digital objects independent of information systems and for enhancing digital objects with ancestry information. The ancestry information allows the identification of digital objects used in the creation of the digital object and its ancestors. We discuss the existing definition of a digital object, its equality relationships, and its identification through intrinsic properties. For the representation of ancestry relationships we propose the application of the graph theory and discuss possibilities of embedding digital object identification in existing identifier systems.

We specify requirements for the information system agnostic integration of digital object ancestry with the digital object and propose a method which implements these requirements. Our method is called SIMPLE (*S*imple *I*dentifiable *M*etadata with *P*ersistent *L*ineage *E*mbedding) and integrates the digital object with its metadata so that these metadata become an integral part of the digital object independent of the information system.

The claims are validated with a prototypical implementation of SIMPLE and its deployment in a case study with the State Archives of Appenzell Ausserrhoden. In this case, the ability to document the ancestry of a digital object results in transparency and traceability in the processing of the digital object.

Table of Contents

1

Introduction

More and more information is exclusively available in digital form. In contrast to information in physical form digital information is highly malleable. Its content may change without leaving any trace of the previous content. Digital information can be copied without any loss of information and the copy is indistinguishable from the original. Additionally, digital information can be merged with and transformed into other digital information. However, the products of these transformations do not normally convey their sources; no records of their ancestry persist.

The lack of standards for the identification of digital information hinders the referencing and citing of sources for a derived digital object. Even if the creator of digital information would like to reference, cite, or attribute his digital sources, there is no recognized way of doing so, particularly if the citing entity provides no referencing means. A paper may attribute its sources. In contrast, a derived data set provides no means for referencing its ancestors or contributing data sets. The decision on how to cite digital sources becomes difficult and troublesome, leading to omission of the attribution. Since attribution is one of the pillars of academic progress, its omission must not be taken lightly; the lack of established standards for the citation of digital information impairs academic progress.

Another pillar of academic progress is the repeatability of experiments, independent of the original experiment. An experiment with the same preconditions and the same process should yield the same results. At present, many experiments consist of an evaluation of data which was generated in previous experiments from a different point of view, and the re-evaluation is highly automated and implemented in computer programs. Those experiments will only remain repeatable if the exact source data can be re-established. For this to be the case, the exact data needs to be identified and located; it is part of the ancestry of the experiment.

1.1 Problem Statement

Which digital objects were used to create the digital object at hand? Today, this question cannot be answered for most digital objects; there is no record of digital object ancestors – no ancestry. The question has three parts, namely the identification of ancestors, the ancestry relationships,

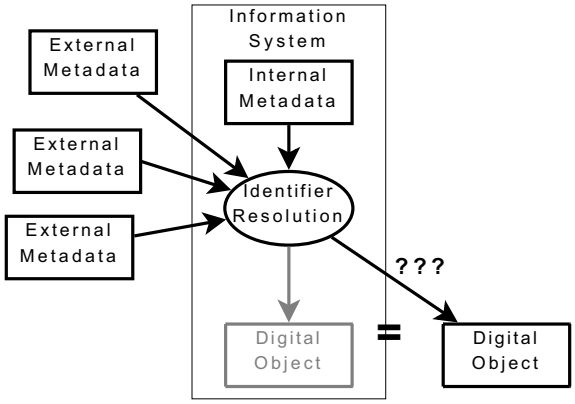


Figure 1.1: The difference between metadata referencing a digital object within an information system and referencing the same digital object outside of the information system

and the accessibility of ancestry. It is clear that ancestry relationships cannot stand without the identification of ancestors, or more specifically, the identification of digital objects. Furthermore, the ancestry has to be accessible to the user of the digital object. How can the ancestry of digital objects be conveyed? Since communication of ancestry to a user depends on the context of the digital object, which is defined by the information system that manages the digital object, the answer to this question is multifold. However, we aim to answer the question as independently of the context as possible.

Ancestry puts a digital object into context. Today, digital objects are put into a larger context by external metadata. In archives, this metadata will be hierarchical, leaning on ISAD(G) [International Council on Archives, 1999].[1] Other organizations and persons have other needs and will collect and organize metadata accordingly. Metadata concerning digital objects will always reference the digital object, thereby identifying the digital object in a given information system such as an archives, the Internet, a file system, or another identifier resolution system. If the identifier resolution system is bound to an information system the digital object can no longer be identified by its associated metadata (figure 1.1) when removed from this identifying information system. We focus on the digital object as sole context in which case the context stems from this digital object independent of the information system.

The digital objects used to create a digital object, its ancestors, are generally not known. It is therefore not possible to consult the ancestors for a comparison with the digital object. The ancestry of the digital object helps to re-establish the original context by referencing other digital objects. This leads directly to the other aspect of the original question; that of digital object identification.

Today, digital objects are commonly identified by string identifiers or access paths (file names

[1] In the area of digital preservation the metadata context models are more complex and are not solved on the basis of metadata alone (see the OAIS reference model [CCSDS, 2002] and the attempt to transform the model into an operational framework in CASPAR [CASPAR project, 2009]).

or URLs for example) that are resolved in an information system which is defined by the context of the digital object. If no special measures like versioning are taken, these identifiers or access paths only identify the current digital object at the specified location which may change with time. Due to the malleability of the digital object, even a static digital object can be changed very easily, very quickly. A change will normally not be noticed by the referencer, and even if it were, the digital object that was originally referenced will no longer be available or findable. Even if copies of originally linked document still existed and were accessible (in the Internet Archive for example), they would not be found, because the identification was bound to the information system. Much the same problem also exists when citing non-paper publications like digital objects or when embedding citations in non-paper publications thereby establishing the publications' ancestry. There is no standard for scholarly identification or citation of or from digital objects. Moreover, even if the identification is present a researcher cannot easily assert whether or not the digital object at hand corresponds to that used in previous experiments and whether or not its content is identical.

Our approach, which we discuss in this thesis, aims to answer the initial question of digital object identification and ancestry, providing a starting point for establishing the context of digital objects. We call our approach *Digital DNA*. There are some assumptions which we consider for our approach:

Digital object centric. We assume that all digital information can be represented as digital objects, as a finite string of bits.

Access to digital objects. We assume that a digital object is retrievable from an information system as a whole.

Digital object long term preservation. We assume that a digital object can be preserved for the long term without being changed (bit level preservation).

With these underlying assumptions we aim to define a method that identifies digital objects and extends them with ancestry. Other approaches exist that aim at capturing and storing the ancestry of digital objects. However, these approaches target digital objects in controlled environments such as database systems, workflow engines, or versioning systems. In contrast, we consider the situation where single digital objects of unknown origin need to be put into a broader context, with the seeding context being the digital object itself. The question of the interpretation of digital objects is not put forward in this thesis. Therefore, any topics related to the interpretation of digital objects are beyond the scope of this thesis. This includes, but is not restricted to, the rendering, the migration, and the assessment of the intelligibility, authenticity, originality, etc. of digital objects.

Consequently, we have identified the following basic requirements for such a method:

Identifiable ancestry. The method must ensure that the digital object and its ancestry are consolidated in a way that the ancestry can be located together with the digital object.

Ancestry extensibility. The method must allow arbitrary ancestry to be included.

Information system agnostic. The result of the method should not depend on specific properties of an information system managing the digital object.

Digital object format agnostic. The method should not impose constraints on the format of the digital object.

Other approaches also capture ancestry, or in the terms used by other authors, "lineage", "provenance", "pedigree", and "processing history" [Bose, 2004, Gladney, 2007, Quenault, 2004, Cundiff, 2004, CCSDS, 2007, 21C, 1997, Spéry et al., 2001]. However, they are based on different assumptions and requirements than ours.

The most crucial difference to our approach is their assumption that the information systems that store digital objects are implicitly stable and trusted information systems. They assume that these systems store digital objects for as long as they are referenced and that they are able to guarantee their integrity. Additionally, they assume that the ancestry of a digital object will be stored in the information system. However, they do not define how this information can be retrieved if a digital object is no longer under the control of the system or if the information system is no longer available.

In our approach the digital object is not bound to any information system, since the identification of the digital object only depends on the digital object itself. Additionally, our approach provides defined guarantees for the integrity of a digital object. As a consequence, no information system is needed to ensure the identification and integrity of a digital object. Therefore, the integrity of a digital object with a known identifier can easily be assessed independently of a specific information system. Based on the identification of digital objects, the ancestors of a digital object can be identified independently of any information system. Additionally, in our approach, the ancestry of a digital object is integrated with the digital object so that it can easily be identified in a format agnostic fashion. In summary, the digital object holds its easily identifiable ancestry, independent of its format or an information system. These characteristics have not yet been considered by other approaches.

1.2 Solution Overview

This thesis consists of the following major parts: a method for the identification of digital objects; a method for representing ancestry; the requirements for the identification of and integration of digital object lineage with digital objects; and the definition of a method that fulfills the requirements defined. We use the term "lineage" for addressing the ancestors of a specific digital object whereas the term "ancestry" is used when addressing all ancestry relationships. In more detail, the major parts are the following:

- *Identification of digital objects.* The identification of digital objects is a prerequisite for ancestry of digital objects. We motivate and present a method for the identification of digital objects using intrinsic properties of the digital object.
- *Ancestry representation.* We motivate and present a model for representing ancestry. We also present transformations of the model to enhance its use. These transformations include the addition and elimination of ancestors.
- *Requirements for lineage integration with digital objects.* We motivate the integration of lineage with digital objects and define its requirements. Furthermore, we illustrate related work and the differences in assumptions and requirements.
- *SIMPLE (Simple Identifiable Metadata with Persistent Lineage Embedding).* We present a method called SIMPLE, which implements the defined requirements. We define a generic extensible data model which represents digital object metadata and its implementation to the bit level. For this thesis we will concentrate on ancestry metadata.

Additionally, a prototype that implements the aforementioned method has been developed to demonstrate the feasibility of the concepts proposed in this thesis.

1.3 Thesis Structure

The rest of this thesis is organized as follows.

Chapter 2 provides an overview of related work and defines the terms and concepts used in the thesis.

Chapter 3 presents a method for the identification of digital objects based on their intrinsic properties and defines the theoretical foundation of ancestry representation.

Chapter 4 defines and motivates the integration of ancestry information with digital objects. We present a method for the integration of ancestry information with digital objects.

In chapter 5, we show a prototypical implementation of the requirements from the previous chapter. A case study of the prototype in use shows the feasibility of our claims.

Chapter 6 highlights the limitations of our approach and proposes future work based on the experiences with this thesis.

Finally, chapter 7 shows the contributions of and summarizes the main findings from this thesis; it ends with an overall conclusion.

2

Definition of Terms and Related Work

The terms used in this thesis have different meanings in various research areas. In this chapter, we define these terms in the context of our work and give a rationale for the definitions. We also show commonly used alternative definitions and highlight the differences between these and ours as well as possible in order to bridge the semantic gap between the definitions.

2.1 Notation

The following notation is used in this thesis as in [Mironov, 2005]:

$\{0,1\}^n$ the set of all binary strings of length n.

$\{0,1\}^*$ the set of all finite binary strings.

$H : A \mapsto B$ function H from set A to set B.

$|w|$ the length of string w.

$w[i]$ the bit value of the ith bit of string w, where $0 \leq i < |w|$.

$w \| v$ concatenation of strings w and v.

2.2 Digital Object

The term "digital object" is used in many research areas with different semantics. The term is popular because the concept of "object" is widely used in computer science (object-oriented programming) and therefore some properties of objects are commonly assumed. The definition of "digital object" varies greatly as shown below. We define the term digital object as follows.

Digital Object: A digital object DO is an empty or finite string of bits.

$$DO \in \{0,1\}^* \tag{2.1}$$

We chose this definition because it is free from interpretation, easily understandable, and is compatible with the other definitions of the term. Now we will highlight other definitions of the term "digital object" and show some possibilities for mapping the different definitions to the one above. Since our definition represents the lowest common denominator, other definitions will contain more metadata on the structuring of the defined digital digital object. This extra metadata does not fit into our digital object definition, but needs to be retained in order to conserve it for possible reversion of the mapping.

The Digital Library Community uses the following definition (and variations thereof):

- A *digital object* is a way of structuring information in digital form, some of which may be metadata and includes a unique identifier. [Arms et al., 1997]

In the definition above, the digital object structures information in a digital form which must include a unique identifier and may include additional metadata. To map such a digital object to our definition the 'digital form" must first be mapped to a binary form. Assuming that the digital form consists of a sequence of digital values ($DO_{Arms} = (d_0, d_1, \ldots, d_n)$), this is done by assigning a number to each possible digital value ($d_i \mapsto\ r_i \in \mathbb{N}_0 \,\forall\, 0 \leq i \leq n$). The number of bits per digital value is determined by the number of bits needed to represent the highest digital value of the digital object ($m = \log_2(\max(DO_{Arms}))$). Each digital value is then converted to a string of bits using the most significant bit (msb) encoding scheme ($d \mapsto\ (b_0, b_1, \ldots, b_m), b_i \in \{0, 1\}$). The resulting bit strings are then merged into one bit string in the form $\{0, 1\}^{n*m}$ in the order given by the original digital object:

$$\begin{aligned} DO_{Arms} = (d_0, d_1, \ldots, d_n) &\mapsto ((b_0, b_1, \ldots, b_m)_0, (b_0, b_1, \ldots, b_m)_1, \ldots, (b_0, b_1, \ldots, b_m)_n) \\ &\mapsto (b_{00}, b_{01}, \ldots, b_{0m}, b_{10}, b_{11}, \ldots, b_{1m}, \ldots, b_{n0}, b_{n1}, \ldots, b_{nm}), m = \log_2(\max(DO_{Arms})) \end{aligned} \quad (2.2)$$

The *Data Dictionary for Preservation Metadata* [PREMIS Working Group, 2005] is based on the definition by [Arms et al., 1997]. It features the following definition: "An *Object*, or *Digital Object*, is a discrete unit of information in digital form". The digital object definition names three subtypes (file, bitstream, and representation), defined here to better show the ideas behind the PREMIS (digital) object:

- A *file* is a named and ordered sequence of bytes that is known by an operating system. A file can be zero or more bytes and has a file format, access permissions, and file system statistics such as size and last modification date.
- A *bitstream* is contiguous or non-contiguous data within a file that has meaningful common properties for preservation purposes. A bitstream cannot be transformed into a standalone file without the addition of file structure (headers, etc.) and/or reformatting the bitstream to comply with some particular file format.

- A *representation* is the set of files, including structural metadata, needed for a complete and reasonable rendition of an Intellectual Entity. For example, a journal article may be complete in one PDF file; this single file constitutes the representation. Another journal article may consist of one SGML file and two image files; these three files constitute the representation. A third article may be represented by one TIFF image for each of 12 pages plus an XML file of structural metadata showing the order of the pages; these 13 files constitute the representation.

The definitions above show the broad range of semantics the term digital object can be applied to. They postulate a specific environment (e.g. an operating system using access permissions and file system statistics) and place restrictions on definitions ("a bitstream cannot be transformed into a standalone file").[1] It is therefore difficult to map the PREMIS definitions to our definition, particularly since the definitions address different levels of aggregation (bytes, bits, and files). However, there is no contradiction between the PREMIS definitions and our definition. The PREMIS (digital) object defined as "unit of information in digital form" is both more constricted, since it defines an object as a "unit of information" and more open, as the object can be of any digital form. Therefore, for a PREMIS object to become a digital object as defined in this thesis its digital form has to be converted into a string of bits. The digital form is translated into binary form as discussed above. We will also define mappings for the concrete subtypes of the PREMIS digital object.

A file defined as a sequence of bytes ($file_{PREMIS} = (B_0, B_1, \ldots, B_n)$) is mapped by converting each byte into a string of bits and then merging the mapped bytes in the order of the original byte sequence ($B = (b_7, b_6, \ldots, b_0), b_i \in \{0, 1\}$). To enable the mapping of the bytes, the order of the bits needs to be defined. This can either be done in the most significant bit (msb) or the least significant bit (lsb) order. This additional metadata is needed for the reversal of the mapping. The same applies to other additional metadata associated to the PREMIS file such as file name, access permissions, etc.

$$\begin{aligned} file_{PREMIS} = (B_0, B_1, \ldots, B_n) &\mapsto ((b_0, b_1, \ldots, b_7)_0, (b_0, b_1, \ldots, b_7)_1, \ldots, (b_0, b_1, \ldots, b_7)_n) \\ &\mapsto (b_{00}, b_{01}, \ldots, b_{07}, b_{10}, b_{11}, \ldots, b_{17}, \ldots, b_{n0}, b_{n1}, \ldots, b_{n7}), b_{ij} \in \{0, 1\} \end{aligned} \quad (2.3)$$

The PREMIS bitstream definition of "contiguous or non-contiguous data within a file" implies that a bitstream is composed of known parts of a file. We can therefore use the same mapping as in the PREMIS file. The PREMIS representation, defined as a "set of files, including structural metadata" adds additional structure metadata on top of the file definition ($Representation_{PREMIS} = (file_{PREMIS_0}, file_{PREMIS_1}, \ldots, file_{PREMIS_n})$). To map a representation to a digital object as defined in this thesis the files, or more specifically, the contents of the files, are merged into one file.

[1] A counter-example is an email attachment. The attachment is a bitstream and a file in itself when extracted and stored in a file system.

This one file can then be converted to a digital object as described above. The order and length of the merged files need to remain available to enable reversal of the mapping. The same applies to other additional metadata associated to the merged PREMIS files as described above for single files.

The *Open Archival Information System Reference Model* (OAIS RM) standard [CCSDS, 2002] is one of the greatest influences on the vocabulary of both the digital archives and libraries communities. As it is an ISO standard (ISO 14721:2003), it influences the IT industry worldwide. The OAIS provides the following definition of a digital object: "An object composed of a set of bit sequences" ($DO_{OAIS} = (bs_0, bs_1, \ldots, bs_n)$). The mapping of our definition to the OAIS is therefore similar to the mapping of the PREMIS file. The set of bit sequences can be converted to a finite string of bits by merging the bit sequences of the OAIS digital object into one string of bits, which is a digital object by our definition. The order and length of the merged bit strings need to remain available to enable reversal of the mapping.

$$\begin{aligned} DO_{OAIS} = (bs_0, bs_1, \ldots, bs_n) &\mapsto \left((b_0, b_1, \ldots, b_m)_0, (b_0, b_1, \ldots, b_p)_1, \ldots, (b_0, b_1, \ldots, b_q)_n\right) \\ &\mapsto (b_{00}, b_{01}, \ldots, b_{0m}, b_{10}, b_{11}, \ldots, b_{1p}, \ldots, b_{n0}, b_{n1}, \ldots, b_{nq}), b_{ij} \in \{0, 1\} \end{aligned} \quad (2.4)$$

It should be noted that while the OAIS RM defines a digital object as a set of bit sequences it often refers to the digital object as "a sequence of bits." This definition corresponds directly to our definition of the digital object and therefore needs no mapping.

In conclusion, we note that in all definitions the notion of an "object" is much more concrete than in computer science. In the object-oriented paradigm, an "object" is an abstract conceptual entity embedded in a broader conceptual framework (lacking in digital preservation). In this thesis the "digital object" is defined as a finite string of bits. This definition was chosen because it represents the lowest common denominator (with respect to the currently used information systems) and the definition is independent of any reference to storage, transfer, processing, interpretation, format, grouping, etc. In contrast to an object in computer science, the digital object does not encapsulate any state nor does it define an interface of methods; it exposes its complete content (its bits).

Digital objects as defined in this thesis are assumed to contain data. However, their data is not inherent to the digital object. The bits of the digital object have to be interpreted to get the data. A string of eight bits with the binary representation of *00101010* (hex: 2a, dec: 42) can be interpreted as the ASCII symbol '*', as the number 42, as a part of the TIFF magic number, or as an instruction to buy all books by Douglas Adams. It depends on how the digital object is interpreted. Digital objects are passive by definition and rely on external interpretation and therefore external activity.

2.3 Digital Object Identifiers

An *identifier* identifies an entity within a given context [Kindberg, 2002]. Entities can be physical objects or abstract concepts. An identifier is a string, a number, a symbol, or combinations thereof. The identifier, "C:\AUTOEXEC.BAT" for example, identifies a file on the hard disk of a computer, while the number 0330258648 identifies a book. However, depending on the context "C:\AUTOEXEC.BAT" could identify files with different content and 0330258648 could be the serial number of a bath towel. The relationship between an identified entity and its identifier is a one-to-many relationship. One entity may be identified by multiple identifiers while an identifier must only identify one identifiable entity.

For this thesis a digital object identifier[2] identifies a single digital object. Therefore, only identifiers able to identify single digital objects will be considered from here on. The International Standard Book Number (ISBN) for example cannot be a digital object identifier since it can only be assigned to monographs. Digital object identifiers must therefore be able to have digital objects as identifiable entities and those must be unique within a given context. Examples that satisfy this property are the Archival Resource Key (ARK) [Kunze and Rodgers, 2007], Handles [Sun et al., 2003], and identifiers defined on that basis (like the Digital Object Identifier (DOI) [Paskin, 2007]), PURL [Shafer et al., 1996], as well as any self-defined identifiers (on the basis of the Uniform Resource Identifiers [Berners-Lee et al., 2005], for example).

Identifiers may be assigned to or derived from the digital object in any stage of its life cycle. They may even outlive the digital object itself.

2.4 Intrinsic Properties

Every digital object has properties that can be observed:

1. The properties of the content of the digital object (its bits).
2. The properties that are extracted through transformation of the content of the digital object.

The properties can be used for distinguishing between different digital objects; they are part of the identification of digital objects.

The properties of the contents are its length (number of bits) and the string of its bits. The extracted properties, which are a super-set of the content properties, are all properties that can be derived by the transformation of the digital object content. Examples for transformations range from the number of ones and zeros in a digital object on the bit level to the description of a

[2]The digital object identifier is not restricted to the digital object identifier (DOI) as used in the digital object identifier system managed by the International DOI Foundation. DOI® is a registered trademark of the International DOI Foundation.

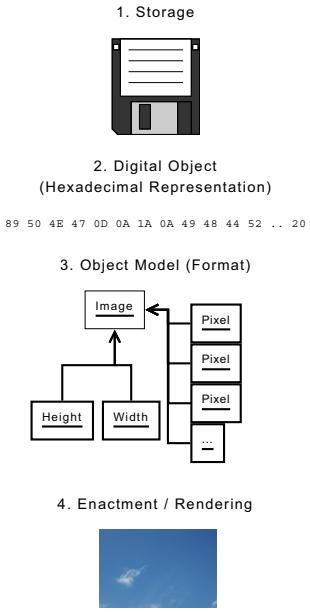


Figure 2.1: Digital object storage to rendering

rendered digital object ("a blue sky"). *Significant properties* such as "the characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects" [Wilson, 2007] or "those properties of digital objects that affect their quality, usability, rendering, and behavior" [Hedstrom and Lee, 2002] can be used in an archival context to describe the "essence" of a digital object, or more precisely, of its rendered form.

Each level of abstraction from the stored digital object to the rendered digital object (figure 2.1) can be the source of intrinsic properties. However, the first level (storage) is not of interest, since the digital object as we have defined it starts at the second level (digital object). On this second level intrinsic properties can be defined using bit-level functions on the digital object ($f : DO \mapsto Properties$). On the third level the digital object has been converted to objects in an object model defined by a format definition and applied to the digital object. On this level the functions can be applied to more complex data like strings, numbers, pixels, etc. The functions used on this level are determined by the objects used in the format, like a color histogram for a digital image. The most primitive properties are the format and the data elements extracted from the digital object as defined by the format. However, the format of the digital object is often not easily determined from the digital object, but provided by the information system controlling the digital object in the form of a file name suffix for example. On the fourth level the objects from the object model are rendered for human reception. The functions on this level are defined by the human mind as a result that can be communicated, for example, a description of a digital image ("a blue sky").

The properties of a digital object that are of the most interest are those that can be extracted directly from the bits of the digital object. A special group of such functions are cryptographic hash

functions ([Naor and Yung, 1989], [Menezes et al., 1997], [Preneel, 1993], [Mironov, 2005], etc). The output of the cryptographic hash functions can be used to identify digital objects and to detect changes in the digital object or parts thereof. [Crespo and Garcia-Molina, 1998] proposed the use of the output of cryptographic hash functions ("signatures") as "object handles" to identify documents within a digital library repository. They also provided a formula to compute the probability of not having "signature collision", p depending on the number of documents n and the number of bits in the signature b:

$$p = \prod_{k=0}^{n-1} \frac{2^b - k}{2^b} \tag{2.5}$$

For practical use, this formula can be approximated by:

$$p \approx e^{-\frac{n(n-1)}{2^{b+1}}} \tag{2.6}$$

[You et al., 2005] proposed a file identification by content which they define as "single continuous stream of binary data". They use the output of a hash function as a primary content address. This approach is also used in content addressed storage systems [EMC Corporation, 2007]. However, [You et al., 2005] proposed to augment the content address with a small amount of metadata to resolve collisions. [Quinlan and Dorward, 2002] proposed an identification of blocks of data using collision resistant hash functions to generate "fingerprints" of the stored blocks. The fingerprints are then used to retrieve blocks. Their formula for the probability of collisions of the identifiers of two data blocks p, assumes random hash values with a uniform distribution and depends on the number of blocks n and the number of bits in the fingerprints b as follows:

$$p \leq \frac{n(n-1)}{2} \times \frac{2}{2^b} \tag{2.7}$$

Using this system, a collision of two identifiers of 128 bit length within an exabyte of data (10^{18} bytes) and a block size of 8 kilobytes ($8 * 10^3$ bytes) would occur with a probability of less than 10^{-20}.

2.5 Digital Object Ancestry

The work in this thesis focuses on the "parentage" of digital objects. More specifically, we want to be able to answer questions about the identity of the predecessors and successors of specific digital objects. We have not found a term in the literature that conveys this notion in the context of digital data. Therefore we use the term *ancestry* which has not yet been used in the literature to the best of our knowledge. The term "lineage", which is commonly used, implies knowledge of the inputs for the creation of a digital object and on the processes, software, and functions used to create that digital object. This includes all processing done on all sources leading to the object

in question. The semantics of the term lineage, contrary to the term digital object, are relatively consistent in different areas of research. Lineage is often used interchangeably with *provenance* and *pedigree*.

We define the term *digital object ancestry* and the closely related terms as follows:

Primal Digital Object: A digital object with no digital object ancestors.

Terminal Digital Object: A digital object with no digital object descendants.

Digital Object Ancestor: A digital object used to create another digital object.

Digital Object Descendant: A digital object with at least one digital object ancestor.

Digital Object Lineage: The directed graph of all digital object ancestors of one terminal digital object.

Digital Object Ancestry: The directed graph of all ancestors and all terminal digital objects.

Note the difference between lineage and ancestry. While lineage (figure 2.3) focuses on *one* terminal digital object, ancestry (figure 2.2) is defined more broadly, incorporating all terminal digital objects to give a view of all connections between the digital objects. Our definition borrows from graph theory and can easily be translated to it. This is described in the following section.

2.5.1 Ancestry Graph

An *ancestry graph* is a *directed graph* with some restrictions. We will first define the directed graph and thereafter the ancestry graph [Diestel, 2005].

A *directed graph* D is a pair (V, E) of disjoint sets (*vertices* V and *edges* E) that has two maps $init : E \to V$ and $ter : E \to V$ which assign to every edge e an *initial vertex* $init(e)$ and an *terminal vertex* $ter(e)$. The edge e is *directed from* $init(e)$ to $ter(e)$.

An *ancestry graph* is a *directed graph* with the following restrictions.

1. No loops: No directed path that includes any vertice v more than once can exist.
2. No multiple edges: The set $(init(e), ter(e))$ must be disjoint for every edge e in E.

Additionally, in ancestry graphs, all vertices are called *digital objects*, the vertices of $init(e)$ are called *ancestors* and those of $ter(e)$ are called *descendants*. These adaptions to the ancestry nomenclature simplify communication and set the boundaries for the discussion. We will use the definitions of terms as laid out in this chapter.

A *lineage graph* L is an ancestry graph A with only one terminal digital object, namely the digital object *do* for which the lineage graph is defined. All directed paths of a lineage graph end

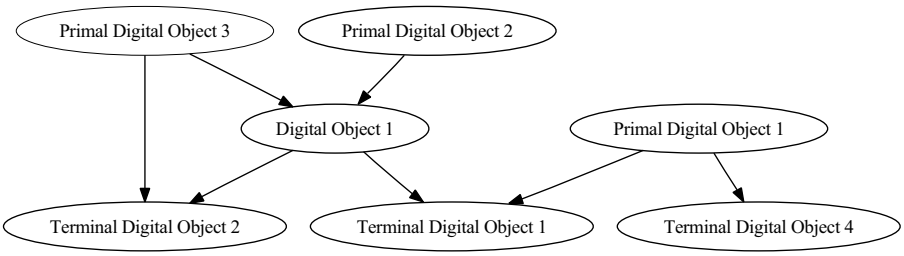


Figure 2.2: Digital object ancestry

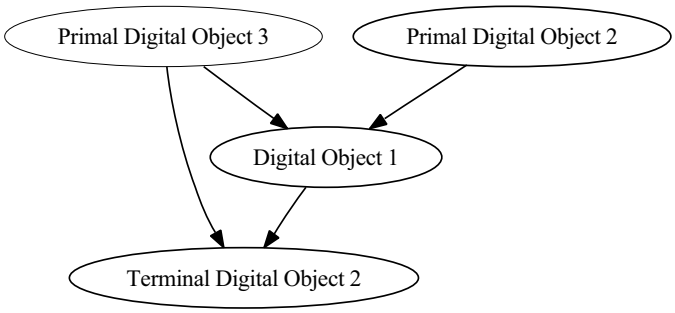


Figure 2.3: Digital object lineage

in the terminal digital object do. This is achieved with function $lineage(A, do) \rightarrow L$. This function recursively removes all terminal digital objects except for the chosen digital object for the lineage, followed by the parts of the graphs not connected to the chosen digital object. Theses steps re-iterated until the chosen digital object is the only primal digital object left in the ancestry graph, resulting in the lineage graph L. We call $lineage(A, do)$ the *lineage function*. Figure 2.4 shows an example of the lineage graph of a digital object.

Our definition of ancestry puts more emphasis on the digital objects and less on the process that created the digital object or the time of their creation. However, this can easily be changed by adding semantics to the edges of the graph. To the best of our knowledge, no standard way to describe the general digital object creation process exists in the literature. The development of a theoretical foundation would go beyond the scope of this thesis, hence we leave this to future work. We will now put our definition of ancestry and lineage into the context of related work.

2.5.2 Related Work

[Bose, 2004] defines *data lineage* as "sources and deviation of a data set or product". It also "encompasses data acquisition and compilation methods, conversions, transformations, and analyses, along with the assumptions and criteria applied at any stage of the data set life cycle." Additionally, "data lineage may also refer to items that have evolved from a data set or product." The

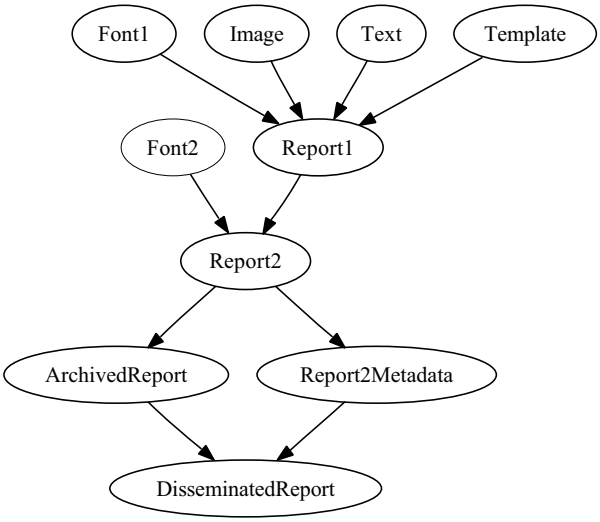


Figure 2.4: An example for a digital object lineage graph

lineage metadata model proposed for this definition is based on the notion of a scientific workflow with *data products* as in- and outputs and *data transformations* that are controlled by a *workflow invocation*. The workflow invocation records all invocations of the software objects (the *data transformations*) and their in- and output as lineage metadata. The workflows are defined by a "directed acyclic graph (DAG) consisting of a succession of alternating *data product* and *data transformation* nodes." In comparison to our definition, the data objects in the work of [Bose, 2004] resemble our digital objects as the basis for lineage. However, since the data product is never formally defined by [Bose, 2004], we cannot provide a conversion mechanism. In contrast to our approach, the data transformations are captured explicitly. The capturing and management of this lineage metadata is part of the role of the workflow engine. Consequently, the system borders are defined by the workflow engine, while our approach is not constrained by such a dependence.

[Widom, 2005] proposes lineage as part of a database management system (Trio), which combines data, accuracy, and lineage in an integrated fashion. Lineage is captured and stored based on records (tuples) where updates, data derivations, bulk data loads, and imports of data from sources outside of the system form the starting points of the lineage. Lineage data is defined by the following relationship: `Lineage(tupleID, derivation-type, time, how-derived, lineage-data)`. The `tupleID` attribute refers to the tuple in the database for which the lineage tuple was created. The attributes `derivation-type` and `how-derived` describe how the tuple was derived. The attribute `lineage-data` describes the data from which the tuple was derived, and the attribute `time` describes when the tuple was derived. In the Trio system, the tuple IDs are never reused and tuples are never deleted or updated. Whenever a delete or update occurs, the tuple in question is "expired" and remains accessible in the system. Similar to other relational database research, lineage in the Trio system is regarded as an intra-system property where only

data within a database system is lineage-enabled. Compared to our definition of lineage, the tuple is the object under lineage control analogous to the digital object. Information on how and when the tuple was derived is also part of the lineage information, which is not the case in our definition.

[Buneman et al., 2000] identified "data provenance" as "the process of tracing and recording the origins of data and its movement between databases" in the context of structured and semi-structured data-sets. They differentiate between a *why* and a *where* provenance; the why-provenance identifies the data that contributed to a resulting datum, while the where-provenance identifies the data that is present in the resulting datum. In a simple example, the where-provenance of the result of $max(\{2, 3, 1\})$ is "the second item of the list $\{2, 3, 1\}$", while the why-provenance includes the whole list because every item of the list contributed to the determination of 3 as the maximum from the list. The provenances are defined in detail in [Buneman et al., 2001]. Additionally, [Buneman et al., 2000] provides insight into the problems of data citation, namely the citation of (parts of) databases and semi-structured data like data in XML form, and the question of the responsibility of maintaining the integrity of cited data source parts and ends with as follows:

> "Finally, one is tempted to speculate that we may need a completely different model of data exchange and databases to characterize and to capture provenance. One could imagine that data is exchanged in packages that are self aware and somehow contain a complete history of how they moved through the system of databases, of how they were constructed, and of how they were changed. The idea is obviously appealing, but whether it can be formulated clearly, let alone be implemented, is an open question." [Buneman et al., 2000]

We aim to give a partial answer to the question posed for our digital objects. We see the digital object as the package that contains its history, albeit not completely. By focusing on the derivation of digital objects (ancestors and their descendants) and masking the processing of the digital objects, the ancestry relationship between the digital objects becomes clearer. This framework may be enhanced by a more complete coverage of the process that leads to the creation of the digital object. This enhancement, however, is left to future work.

[Woodruff and Stonebraker, 1997] define the lineage of a datum as "its entire processing history". This includes the origin of the data as well as all subsequent processing steps. Using this information, combined with the proposed *weak inversion* functions (functions which invert processing steps with defined guarantees), the data that influenced a datum can be retraced. This is equivalent to the where-provenance as defined in [Buneman et al., 2000]. Additionally, the inversion functions are classified as *complete* (yielding no false negative data items) and as *pure* (yielding no false positive data items). The scope of lineage is very fine-grained; the lineage of each datum is assessed separately. The lineage is recorded, while the data used to calculate a

datum is identified upon request by invoking the weak inversion functions of the recorded processing steps. To successfully identify the source data all processing steps must be known and have a weak inversion function assigned to them.

Compared to our definition of lineage, the datum (within a data set) is the object under lineage control. However, the lineage is not recorded, but generated by applying inversion functions to the intermediate products of a process featuring multiple processing steps. To achieve this, all data sets (source, intermediate, and final), as well as all steps and inversion functions, must be known. A communality between our approach and that of [Woodruff and Stonebraker, 1997] is that [Woodruff and Stonebraker, 1997] also suggest a directed acyclic graph (DAG) as the representation of lineage.

[Spéry et al., 2001] proposes the capture of changes in a catastre with its geographical objects in a lineage DAG in which every change in a parcel is motivated by a document. In this way, the lineage of every object can be retraced. The relationships between the ancestor ("parent") and its descendants ("children") are used to characterize the catastral change retrospectively. The definition of lineage corresponds with our definition, as every change leads to a new object.

[Grant, 2009] presents a way for accessing the version control system git [GIT, 2009] via file system operations. This allows to track all changes to files within a directory over time. These changes are stored in the version control system and can be queried. Even though git does not use the term lineage, the changes of files an directories can be traced back in time. The version control system can not identify the source of the changes to a file, for example if some additional text in a file is copied from another file, the version control system will only note the new lines without the information about the ancestor file.

In the research area of archival studies [Duranti, 1998] lays the foundation for the field of diplomatics in six articles published in Archivaria between 1989 and 1992. However, the comparison of ancestry with diplomatics is out of the scope of the thesis and left for future work.

2.6 Metadata Adhesion

In this, section we define and describe *metadata adhesion*. Metadata adhesion is a qualitative measure for the effort needed to dissociate metadata from the digital object. We will consider metadata managed by the same information system as the digital object as well as metadata managed by other information systems. To the best of our knowledge, this association of metadata with the digital object has not been subject to previous research. We will present an overview of the techniques by which metadata has been associated with digital objects to date, and will rate them in terms of metadata adhesion.

Today, metadata concerning the digital object is either packaged with the digital object, or it references the digital object. The base techniques are therefore *referencing* and *packaging*, both of

Reference Types	Explicit	Implicit
Hard	Foreign to primary key relationship in a database management system. File system metadata for a file.	- (We have not found any examples for implicit hard references.)
Soft	Hyperlink in a HTML document.	Checksum Files [Williamson et al., 2002]. Sidecar files [Adobe, 2005].

Table 2.1: Examples for reference types

which we will discuss in the remainder of this section.

2.6.1 Referencing

When metadata references a digital object this reference is either *explicit* (providing an unambiguous reference to the digital object) or it is *implicit* (relying on an (system-)immanent reference). Additionally, the references are distinguished by the nature of their management. References with integrity constraints that are actively enforced by the information system managing the metadata which references the digital object are called *hard* references. In contrast, references with no integrity constraints or no active enforcement are called *soft* references. Soft references may be *dangling* if the referenced digital object cannot be retrieved. See Table 2.1 for examples for the different types of references.

For explicit hard references, the metadata holds a reference to the digital object it describes. The reference is defined specific to the information system which is responsible for managing both the digital object as and its metadata. Any change in the digital object reference will be reflected in the reference of the metadata. Since the metadata is closely linked to the digital object, the deletion of the digital object must trigger the deletion of its reference, for example by deleting the metadata as well. The metadata adhesion for explicit hard links is high since reference between the metadata and the digital object is enforced by an information system.

For implicit hard references, the references between the digital object and its metadata are implicitly defined in the the information system which manages the digital object and its metadata. We do not know of any information systems that use implicit hard references.

For explicit soft references, the metadata will generally not be managed by the same information system as the reference. Therefore, the reference needs to provide some basic, system independent reference information. The reference between the digital object and its metadata is governed by a naming convention and a name resolution service which resolves a given name to a digital object (association by identifier or name). It is assumed that the information system which manages the metadata can only access the digital object using the supplied reference. The reference to the digital object is established by an unambiguous identifier of the associated digital object.

If, for example, a file is renamed, its identifier is changed, and no referential integrity checks are performed to ensure that the file can be renamed without affecting references to it. Any access to the file thereafter will not be informed of the new name, and no support is available to locate the previous file within the information system.

The metadata adhesion for explicit soft references is medium, since the reference of the metadata to the digital object will not be updated if the identifier changes or the digital object is deleted, leaving the reference in a dangling state. The metadata adhesion is not low because the reference is available, which helps in locating the referenced digital object, even if its reference has changed.

For implicit soft references, the references between the digital object and its metadata are implicitly defined. The reference is not monitored or enforced by any information system. If the digital object is relocated or renamed the reference will be dangling. The metadata and the digital object are normally managed by the same information system, even though this system may not be aware of the reference. This is the case for checksum files that provide integrity metadata for files. The checksum files have the same file name as the file for which they provide integrity information, with an additional extension providing the checksum type, for example `.md5` or `.sha1`. Another example of implicit soft references is the "Software-Invariant Archiving of Relational Databases" (SIARD) archive format, which defines a directory structure for the data and the metadata files generated by the archiving of a database [Heuscher et al., 2004].

The metadata adhesion of implicit soft references is low, since the reference of the metadata to the digital object will not be updated if the identifier changes or the digital object is deleted leaving the reference in a dangling state. The metadata adhesion is lower than that of explicit soft references, because the reference is only available if its definition is known.

2.6.2 Packaging

Packaging binds the metadata of a digital object to the digital object by providing a wrapper that includes both metadata and the digital object. The resulting package is also a digital object. Contrary to referencing, a package will always include the metadata and the digital object. The metadata adhesion for packaging is high since the digital object and its metadata are fused together in one package. The notion that the metadata and the digital object can be retrieved unchanged from the package is implied.

The problem of binding metadata to a digital object has been discussed widely within the digital archives and libraries community, resulting in the proposition of encapsulation, which is the only solution to date. Encapsulation is used to store the digital object and its metadata in one file. For this, the digital object and its metadata need to be reformatted according to the encapsulation file format. The encapsulation formats commonly used are ZIP and XML.

All approaches rely either on the ability of the future user to decode a specific encoding format (for example [LOC, 2005] and [Heslop et al., 2002]), or a functioning information system to link

the metadata to the digital object (for example the OAIS RM standard[3]). These approaches implicitly assume that all metadata is stored outside the digital object, which is not the case for most born-digital objects (JPEG or TIFF images, Microsoft Word documents, etc). Many file formats embed a wide range of metadata which is normally associated with the creation of the digital object [Duval et al., 2002] (for example the last printing date of a document or the exposure time used when taking a digital photo).

Metadata can be packaged with one or more digital objects using either *encapsulation* or *integration*. Encapsulation means that the digital object and its metadata are merged in a single digital object. This is achieved by encoding the digital object and its metadata in a form which is independent of the format of the digital object. The method is implemented by the METS metadata schema [LOC, 2005], the XML Formatted Data Unit (XFDU) [CCSDS, 2008], the Victorian Electronic Records Strategy [Francis et al., 1998], and the National Archives of Australia [Waugh et al., 2000]. Its use was also recommended by [Gladney, 2004] and [Boudrez, 2005]. Integration of metadata means that the metadata is included in the digital object, retaining its format. Most digital object formats support the integration of format specific metadata. Arbitrary metadata, however, can often not be included within the digital object format. However, XMP [Adobe, 2005] can be used as a container for such arbitrary metadata.

The Victorian Electronic Records Strategy and [Boudrez, 2005] encapsulate the digital object in an XML document. They only allow the addition of standardized XML metadata as opposed to arbitrary XML data. The same holds true for the National Archives of Australia, even though it does not define any non-technical metadata in its digital object encapsulation format. The digital object is either $Base_{64}$ encoded or formatted to suit the hierarchical XML organization. XML Formatted Data Units provide the same functionality but they also allow the referencing of metadata as well as the referencing of additional digital objects from the package.

[Gladney, 2004] describes a *trusted digital object* (TDO), which is comprised of an arbitrary number of digital objects and a protection block. The protection block includes an identifier block that uniquely identifies the TDO, a MAC (message authentication code) description block that ensures the integrity of the TDO by cryptographic measures (for signing), a manifest (per digital object) that describes the digital object, a relationship block that describes the relationship between the included or referenced digital objects (or parts thereof), and additional structured and unstructured metadata. No specific form of packaging is provided for the creation of the TDO. The author claims, however, that metadata-to-object dissociation will only occur rarely which leads us to believe that a TDO is meant to be a physical package.

We will now provide a more in-depth analysis of METS in the following paragraph, as it is an established standard which specifically addresses lineage issues.

[3]While the standard [CCSDS, 2002] states that it does not suggest an implementation its mention of information *packages* can be interpreted as a tendency towards encapsulation. First implementations like [DSpace, 2008] or [e-Depot, 2008] all feature encapsulation as packaging strategy.

Metadata Encoding and Transmission Standard (METS) The aims of METS are best described by its *Primer and Reference Manual* [METS Editorial Board, 2006, p. 3]:

> Many institutions in the digital arena are finding it at least desirable if not necessary to maintain metadata about the digital objects they are creating and/or keeping for the long term. Particularly as the complexity and number of digital objects increases these institutions are finding that the metadata necessary for successful management, access, and use of digital objects is both more extensive and different from that metadata used for managing other kinds of collections including physical collections. Many institutions are finding it necessary for instance to retain structural metadata that describes anchors and organizes the components of a digital object together so that the integrity of the digital object may be retained even when its components are stored in different places. And when a repository of digital objects intends to share metadata about a digital object or the object itself with another repository or with a tool meant to render and display the object the use of a common data transfer syntax between repositories and between tools greatly improves the facility and efficiency with which the transactions can occur. METS was created and designed to provide a relatively easy format for these kinds of activities during the lifecycle of the digital object.

The applicability of METS is therefore broad, with the focus on providing containers for data and metadata which are organized by structural metadata. Each METS entity represents a digital object which includes all known or needed lineage information and digital object versions. A digital object version is comprised of one or more files which are either referenced or encapsulated into the METS data.

On the implementation level, one METS entity (one METS XML document) is divided into a METS header, a descriptive and an administrative metadata section, a file section, a structural map section, a structural link section, and a behavior section. We focus on the file and the structural section. The other parts of the METS XML document provide additional metadata on the digital objects which is identified by the file and structural sections. It must be noted that all sections mentioned may also be extended with additional (non-METS) metadata and that this data may be different from institution to institution. METS offers a container for a digital object along with a high-level structure for the metadata of this object. The specific metadata is defined by the METS user. METS provides several examples for every type of metadata.

The METS *file section* identifies or contains all files and identifiable parts of files (for example streams or files contained in a TAR or ZIP file) that make up the electronic versions of the digital object (adapted from [METS Editorial Board, 2006, p. 16]). The basic file and integrity metadata for every file, or part thereof, is stored in this file section.

The METS *structural map section* "outlines a hierarchical structure for the digital object, and links the elements of that structure to content files and metadata that pertain to each element"

[METS Editorial Board, 2006, p. 16]. This section structures the digital content and its accompanying metadata in the METS XML document. It structures and relates the digital content of the file section to the metadata defined elsewhere within the document. The different versions of the digital objects are linked together as well as the metadata for either the files or the digital object.

To conclude, METS offers a core framework functionality that can easily be extended to meet the users' needs. METS focuses on providing a structure for digital content and its metadata to define versions of a digital object. METS does not explicitly address the issue of data - metadata separation, although it provides the means of embedding the digital content which it structures or describes within the METS XML document itself. This is a basic requirement for METS since it would not be possible to transmit (the letter 'T' in METS) digital content without encapsulating it. The embedded files are encoded using the technologies XML provides (for example $Base_{64}$ encoding). Therefore, the content of the file is not directly accessible. Lineage is addressed by METS by defining source and digital provenance metadata in its data structure. Source metadata is data on the analog source of the digital content. It "records descriptive, technical or rights information about an analog source document used to generate the digital library object" [METS Editorial Board, 2006, p. 11]. Digital provenance metadata "records master/derivative relationships between various files which currently represent the object, as well as recording any transformations or migrations undergone by those files subsequent to the initial digitization of an item or, in the case of born digital materials, the files' creation" [METS Editorial Board, 2006, p. 27]. METS records all lineage metadata in one document and then links it to a file or a file group (via the structural map metadata). This approach provides a clear link between the different versions of the files. The ancestry of a file (or part thereof) is directly contained in the structural map metadata of METS.

Forms of Packaging In this paragraph, we define and discuss the two basic forms of packaging, *integration packaging* and *encapsulation packaging*, and introduce their sub-forms. Integration packaging is divided into the sub-forms: *format specific embedding* and *binary append*. Format specific embedding embeds arbitrary metadata within the digital object in a format dependent fashion. The metadata needs to be adapted to the restrictions enforced by the digital object format. This can either be done by recoding the metadata, or by adaptive encoding of the metadata. Additionally, parts of the metadata which can be mapped to format specific metadata can be embedded in the format specific metadata, like the GPS-coordinates in a digital image. Binary append format specifically adds the metadata at the end of the digital object. It is a sub form of format specific embedding, since it restricts the format specific embedding in the location of the metadata embedding. We did not find any sub forms of encapsulation packaging. We will now define the forms of packaging presented and thereafter discuss the differences between them.

Encapsulation: Generating a digital object DO' which contains the metadata MD and the digital

object DO in a suitable form.

$$DO' = encapsulate(DO, MD), Format(DO') \neq Format(DO) \tag{2.8}$$

Format Specific Embedding: Integrating the metadata MD with the digital object DO so that the metadata becomes part of the digital object. The digital object retains its format ($Format(DO)$) and becomes a package DO'.

$$DO' = embed(DO, MD), Format(DO') = Format(DO) \tag{2.9}$$

Binary Append: Appending the metadata MD format specifically to the digital object DO. The digital object retains its format ($Format(DO)$) and becomes a package DO'.

$$DO' = DO \| encode(Format(DO), MD), Format(DO') = Format(DO), \\ DO'[i] = DO[i] \; \forall \; 0 \leq i < |DO| \tag{2.10}$$

The following example which shows the differences between the forms of packaging. Arbitrary metadata is added to a TIFF file in as follows:

- *Encapsulation:* The metadata is stored in a metadata file. The metadata and the TIFF file are packaged into a new ZIP file which contains both files. Both files can only be accessed, if the package structure and format is known.
- *Format Specific Embedding:* The metadata is stored in the comment section of the TIFF file, thereby creating a new version of the TIFF file with a changed internal object model. The metadata is accessed using standard TIFF tools.
- *Binary Append:* The metadata is appended at the end of the TIFF file, thereby creating a new version of the TIFF file with an unchanged object model. For the metadata to be accessed, the location of the metadata must be known.

We will discuss the differences between the packaging forms in view of the following qualitative criteria:

1. *Reversibility* of the packaging process
2. *Transparency* for applications processing the digital object
3. *Complexity* of the packaging and un-packaging process
4. *metadata adhesion* in view of the further processing of the package or the digital object

The criteria have been selected to evaluate the interference of the packaging form with existing digital object processing infrastructure.

The choice of the package format used in encapsulation is free, as long as the digital object and the metadata can be extracted from the package unchanged. The encapsulation can be undone by a reverse function $encapsulate^{-1}$ of equation 2.8: $(DO, MD) = encapsulate^{-1}(DO')$. Neither the package, nor the encapsulated digital object can be handled by software that was used to process the digital object; the digital object needs to be extracted from the package first and therefore the transparency is low. The complexity of encapsulation is low since all digital objects can be encapsulated in the same manner; there is no dependence to the digital object format. The metadata adhesion for further processing is low since the digital object, once extracted, will lose its reference to the package and therefore its metadata.

For format specific embedding, the package format is defined by the digital object packaged with the metadata. The metadata is integrated with the digital object in such a way that it becomes part of a new digital object which retains the format of the previous digital object. As with encapsulation a reverse function $embed^{-1}$ of equation 2.9 can be used to restore the original digital object ($(DO, MD) = embed^{-1}(DO')$). The complexity of the packaging and un-packaging process is greater than that of encapsulation, since the packaging is specific for the format of the digital object. Additionally, the complexity may rise if the digital object has a complex format. However, since the digital object format has not changed there is no need to undo the packaging process for further processing, therefore the process has a high transparency. The metadata adhesion is high for future processing steps since the metadata will still be integrated with the digital object. However, the structure of the resulting digital object package will differ from the original structure of the digital object.

Binary append is a form of format specific embedding, and the conclusions drawn above for this process hold as well. However, some differences need to be observed. First, the reversibility is higher, since the digital object is retrieved from the digital object package by simply ignoring any bits after the initial length of the digital object: $DO[i] = DO'[i] \; \forall \; 0 \leq i < |DO|$ (the reverse function for equation 2.10). The complexity of packaging the digital object and its metadata may be lower for binary append since the location of the metadata is predetermined, limiting the options of metadata integration, specifically the updating of metadata supported by the digital object format. Metadata adhesion may be lower than for format specific embedding since the metadata is not as immersed in the digital object. Hence, applications processing the digital object might ignore the additional metadata. However the metadata adhesion is higher than for encapsulation since the packaged digital object can directly be processed without separating the metadata from the digital object, leading to a high transparency.

2.7 Summary and Discussion

In this chapter, we have established the foundation for the terms and theory used in this thesis and put it into the context of related work. More specifically digital objects and methods on how to identify them have been presented as a basis for creating a digital object ancestry. Furthermore, we have defined the ancestry graph as a tool for defining ancestry relationships between digital objects independently of the information systems which manage them. Finally, we have defined metadata adhesion as a measure for the strength of the bond between a digital object and its metadata and have categorized the various forms of metadata and digital object integration.

In our summary of related work, we found that the term "digital object" has a variety of definitions and different levels of abstraction. However, we could transform these to our definition. Identification of digital objects using their properties as identifiers has been proposed previously and is used in the IT industry (content addressed storage) although tied to a single information system. The restriction to a single information system can be found in the research of lineage, which focuses on the fine-grained capturing or tracing of data from a result back to the source data. The borders of the information system also form the boundary of lineage capturing and interpretation whether it is a database management system or a workflow system. Lineage is recorded on the basis of actions performed on the object under lineage control where the result of the actions lead to different, altered instances of the object.

In the next chapter, we present a method for the identification of digital objects on the basis of their intrinsic properties and define the theoretical foundation of ancestry representation and manipulation.

3

Identification and Ancestry

In this chapter, we present a method for the identification of digital objects on the basis of their intrinsic properties and defines the theoretical foundation of ancestry representation and manipulation.

3.1 Digital Object Identification

This section describes how digital objects can be identified.

3.1.1 Equality

A prerequisite for identification is the definition of equality between digital objects. Equality can be defined at any level of abstraction. However, in order to conform with our definition of a digital object, we define equality on the bit level.

Digital Object Equality: Digital objects are equal, if all their bits are the same.

$$(|DO| = |DO'|) \wedge (DO[i] = DO'[i] \; \forall \; 0 \leq i < |DO|) \rightarrow DO = DO' \tag{3.1}$$

In our definition, the context of the digital object is not present, and therefore the equality is context-free, and depends only on the content of the digital object. A copy of a digital object is therefore the same as the original digital object, even if the contexts are different. The interpretation of the digital object is not considered for equality. Using the definition above, the equality of two separate digital objects can easily be tested; by comparing them by length and content (bitwise). Equality needs to be defined strictly because of the underlying premise that the digital object conforms to an unknown format. Two equal digital objects must not deviate even one bit, as the consequences of any change cannot be assessed for unknown formats.

Context-free equality can be defined differently, using intrinsic properties (subsection 2.4) or the results of functions on the digital object. Equality can be defined in relation to the purpose

and format of a digital object. For example, two digital objects in XML format could both first be canonicalized and then compared. However, such a definition is only valid under the premise that the digital objects have a defined format, which is not the case in this thesis. The question of how to define equality for digital objects is well suited for future research.

3.1.2 Identifiers

If digital object identifiers must be the same for equal digital objects, the identification of the digital object must also be context-free, since equality is defined in a context-free way. Therefore, identification can not rely on the context of a digital object, for example the information system managing the digital object. The digital object identifier must depend on the content of the digital object and must be the result of a function on the digital object, which is an intrinsic property.

We define the digital object identifier as follows.

Digital Object Identifier: A digital object identifier is the result of an identification function ID on the digital object that must ensure that equal digital objects share the same result.

$$DO = DO' \rightarrow ID(DO) = ID(DO') \tag{3.2}$$

The identification function depends on the equality definition for the "=" part of our definition above. The identifier definition logically implies that if two identifiers are not equal, the digital objects identified cannot be equal ($ID(DO) \neq ID(DO') \rightarrow DO \neq DO'$).

We call identifiers that are created with an ID function *intrinsic identifiers*. Consequently, we call identifiers that are externally assigned to a digital objects *extrinsic identifiers*. In practice, identifiers are seldom created without context; the identifiers are created arbitrarily and then assigned to digital objects. These identifiers are created independently of the digital object and are assigned within a defined identifier system to a system or user-defined abstraction. The abstraction may span multiple digital or physical objects as well as their rendering. An example of an identifier is the ISBN number of a monographic publication. This identifier will identify the whole of one monographic publication in various forms. It does not identify a single book which could for example feature an autograph of the author. Translated to the digital environment this means that the different formats of publications would be indistinguishable by their identifier. A digital edition of the publication will have the same identifier whether it is served by a web server as an HTML document or as a PDF document.

The main difference between intrinsic and extrinsic identifiers is the connection between the identifier and the digital object. An intrinsic identifier id_i can be calculated from the digital object using an ID function, ($ID(DO) \rightarrow id_i$). This is not possible with extrinsic identifiers. An extrinsic identifier id_e acts as a link to the digital object. It is the input for a retrieval function R that provides the digital object for the identifier ($R(id_e) \rightarrow DO$). However, the difference is

not always clear-cut, as intrinsic identifiers can be used to retrieve digital objects, while extrinsic identifiers can be embedded in digital objects. Using a custom identifier function ID_e the embedded external identifier id_e becomes an intrinsic identifier and can thereafter be calculated from the digital object ($ID_e(DO) \rightarrow id_e$).

An example for of identification function is the length function $|DO|$, which returns the number of bits in a digital object. The proof that the length function is an identification function is simple; the lengths of two identical digital objects are the same (equation 3.2) and two digital objects that are not of the same length violate the equality definition $|DO| = |DO'|$ (equation 3.1).

Another example of identification functions are hash functions. [Menezes et al., 1997] defines "a hash function h maps bit-strings of arbitrary finite length to strings of fixed length." Translated to our defined terms, a "string" is equivalent to a "digital object." Therefore, in this thesis, a hash function h maps digital objects of arbitrary finite length to digital objects of fixed length. It is implied that the function is deterministic. The proof that a hash function satisfies the requirements of an ID function is as follows: A hash function given two identical digital objects as input will yield the same result (equation 3.2).

Other intrinsic properties of digital objects could even include an externally generated identifier. The digital objects would need to be created to include the identifier. This identifier could be registered centrally or self-defined. For the embedding of the identifier, a header field of a digital object format could, for example be used. We will elaborate on the embedding of information in digital objects in subsection 4.2.3.

Additional requirements on the identification function may add desirable properties to the identifier-to-digital object connection. The most common properties are uniqueness and integrity, which will be discussed in the next subsection.

3.1.3 Uniqueness and Integrity

Uniqueness ensures that different digital objects have different identifiers. *Integrity* ensures that changes to the digital object can be detected. Changes may be intentional, for example updating of digital object, or unintentional, resulting from a transmission error, or malicious, for example intentional updating of the content of the digital object to attain an unfair advantage.

Uniqueness and integrity on a specific ID function both require that different identifiers must identify different digital objects. Such a strict requirement would only be satisfiable if we renounce on context-free equality or, more trivially, if the identifier is the digital object itself. This shortcoming is circumvented in the literature by adding a probability for the upholding of the requirement (see subsection 2.4), which would then read as "different digital object must have different identifiers with a probability of p." We will disclose that probability as a property of the identification function.

For an identifier to enable uniqueness, the identifier has to incorporate uniqueness guarantees. This can be done by adding the requirement that different digital objects must have different identifiers: $DO \neq DO' \rightarrow ID(DO) \neq ID(DO')$. However, as described above, this can only be satisfied if we renounced on context-free equality or, more trivially, if the identifier is the digital object itself. If we relax the requirement in such a way that identifiers from different digital objects differ with a high enough probability p, then the uniqueness problem turns into a well-researched problem of collision-resistance in cryptography. In a controlled environment, the uniqueness requirement could be guaranteed by an information system, for example by using uniqueness constraints on a tuple in a database management system. However, outside of controlled environments, uniqueness cannot be guaranteed.

The requirement of integrity is even closer to the requirement of collision-resistance of cryptographic hash functions h than that of uniqueness. Both require that it should be difficult to mistake a digital object DO' for a given digital object DO that has the same identifier, i.e., $DO \neq DO' \rightarrow h(DO) = h(DO')$ has to be difficult to establish. For integrity to be checked using an identifier, the identifier function needs to show collision-resistance. If integrity checking is a requirement, we use appropriate cryptographic hash functions as identifier functions since they are well researched and well understood.

However, uniqueness and integrity cannot be guaranteed outside of controlled environments, which means that collisions (different digital objects that share an identifier) will occur. Depending on the goal of the identification function and its underlying definition of equality, this can be seen as a bug or as a feature. If equality is based on intrinsic properties, like, for example, a header defining the DOI®, then digital objects with different content will provide the same identifier. The probability of collisions depends on the identification function. The handling or avoidance of collisions is beyond the scope of this thesis.

3.1.4 Embedding Identifiers in Existing Identifier Systems

For an identifier to be of any use, it needs to be embedded in an identifier system. It would be an option to create a new identifier system. We chose not to pursue this option, since it is beyond the scope of this thesis. Instead, we chose to use existing identifier systems, which makes it possible to evaluate the identifier system best suited for embedding and takes advantage of existing infrastructure. Examples of systems the include International Standard Book Number (ISBN), the handle system on which the digital object identifier (DOI) is built, the domain name system (DNS), the Uniform Resource Identifier (URI), or bibliographic identification (as in chapter Bibliography). Not all of the identifier systems mentioned are good candidates for embedding identifiers of digital objects. In this subsection we will provide criteria for identifier systems that show their extensibility.

Clearly, there needs to be a interdependence between the identifier function, its output, and the identifier system. Assuming that the output of the identifier function is again a digital object,

this needs to be rendered to a form which complies with the identifier system. Another option is to design a function that outputs its result in a format that can be used by an identifier system. An example of such a function would be a constrained length function for embedding into an ISBN. The ID function would look something like this: $ID_{ISBN}(DO) = P\|\|DO| \mod 10^m\|C$, where P is the publisher prefix that has been assigned by an ISBN agency, m is the number of digits left in the 13 digit ISBN subtracting the digits of P and C, C is the check digit, and $\|$ is used as concatenation function (Example: $1\|2 = 12$). However, as a precondition for the ID_{ISBN} function to conform to our definition of an identification function, the publisher prefix P must be known. Since the maximum of m would currently be 7 (depending on the publisher), and the length function $|DO|$ will generate identical results (collisions) for different digital objects, the identifiers of such a function would not be unique or collision resistant. Additionally, the ISBN (as defined by its managing entities) was designed to reference monographic publications, not digital objects.

Another example for an embedded identifier would be an identifier embedded in the bibliographic entry for reference. Here, [Altman and King, 2007] proposed a method which allows for the citation of quantitative data within a bibliographic entry by adding a managed identifier and a "universal numeric fingerprint (UNF)." The UNF is an identification function which translates the quantitative data into a canonical form and then applies a cryptographic hash function; this intermediate result is then $Base_{64}$ encoded. Bibliographic references are easily extended with additional information (identifiers in our case), the only restriction being the use of printable characters. Therefore, the requirements of uniqueness and integrity can be embedded easily, for example by an $ID_{bibliographic}$ function that calculates the SHA1 hash of a digital object and converts it to a hexadecimal number, also known as $Base_{16}$ encoding ($ID_{bibliographic}(DO) = Base_{16}(SHA1(DO))$). The result of the function is a string of 40 numbers (0 to 9) and characters (*a* to *f*), *b9f2c814655845fb2495aabbec591b1d1867c96b*, for example. This identifier can be added to any bibliographic reference, aiding the identification of the referenced digital object.

Identifier System Embedding Prerequisites For an identifier system to allow the embedding of custom identifiers for digital objects, the identifier system must allow for an identifier or part thereof to be created independently of the identifier system. The identifier must then be adapted to match the format of the identifier system. In an extensible identifier system like the bibliographic reference system, the identifier must be converted to printable characters. In a closed system like the ISBN system, the identifier must be converted to a (short) string of numbers. Depending on the identifier function of the target identifier system, it may not be possible to embed an identifier in the target identifier system. If for example *SHA1* is chosen as an identifier function, it cannot be embedded into the ISBN system, as the 160 bit of an identifier cannot be fitted into the independently selectable part of the identifier (a maximum of 24 bit).

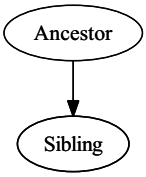


Figure 3.1: Ancestry graph with one ancestor and one descendant

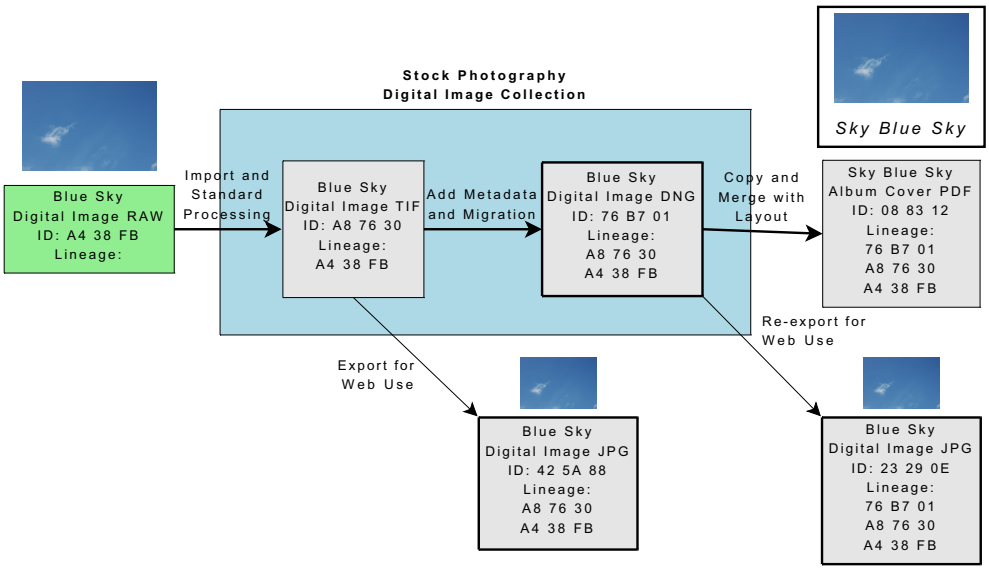


Figure 3.2: An example of the workflow and resulting ancestry graph in a stock photography company

3.2 Digital Object Ancestry

This section outlines and motivates the digital object ancestry view taken in this thesis. We use the term *ancestry* for all of the ancestors and descendants of digital objects, to avoid confusion with lineage. The term *lineage* (of a specific digital object) only includes a subset of the ancestry, namely the ancestors of the digital object.

3.2.1 Ancestry Representation

Ancestry is represented as an ancestry graph (as defined in subsection 2.5.1), like a family tree. An ancestry graph is a specialization of a directed graph with digital objects as vertices and the ancestor-descendant relation as the directed edge, oriented from the ancestor to the descendant vertice (figure 3.1). An example of an ancestry graph is shown in the simple imagined workflow in a stock photography company, in figure 3.2.

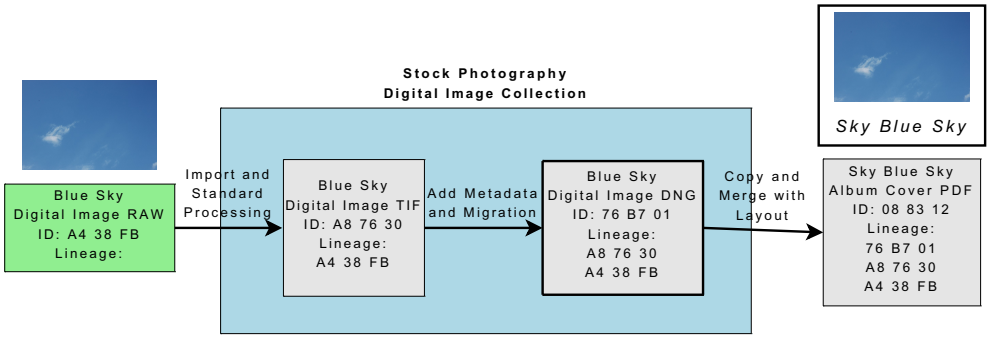


Figure 3.3: Lineage graph for the digital object 08 83 12

A specialization of the ancestry graph is the lineage graph (subsection 2.5.1). This graph only represents the lineage of a defined digital object, which is the only terminal digital object in the graph. Figure 3.1 represents a lineage graph where the descendant is the digital object to which the lineage graph corresponds; the application of the lineage function on the ancestry graph for a defined digital object. Figure 3.3 depicts the lineage graph of the PDF album cover, the digital object *08 83 12*, when using the example ancestry graph as the initial graph (figure 3.2).

3.2.2 Ancestry Manipulation

We call changes to the ancestry graph *ancestry manipulation*. Ancestry can be reduced to satisfy user privacy needs, protection of business transactions and internal workflows, smaller ancestry overhead, or to hide unnecessary detail. On the other hand, the ancestry graph can be expanded to reflect additions to the ancestry, be it by additional research or the undoing of removals in the graph. We distinguish between two forms of ancestry manipulation; *ancestry flattening*, which removes digital objects from an ancestry graph, and *ancestry refinement*, which adds digital objects and ancestry relations to an ancestry graph. A special form of ancestry manipulation is the conversion of a generic ancestry graph into a lineage graph. This conversion has been discussed above and, as all lineage graphs are ancestry graphs, the ancestry manipulations are the same for lineage graphs. In this subsection we will define, motivate and describe the aforementioned ancestry manipulations. We use the functions defined in subsection 2.5.1.

Ancestry Flattening Ancestry flattening is used to remove superfluous or sensitive ancestry parts, namely digital objects, while retaining the structure of the ancestry graph. In an industry setting, it might not be appropriate to include the references to all digital objects used to create a digital object (a digital magazine, for example), since the recipient might receive more information about the digital object than intended by its creator. For example, the creator might not want the recipient to know what images were copied from external sources.

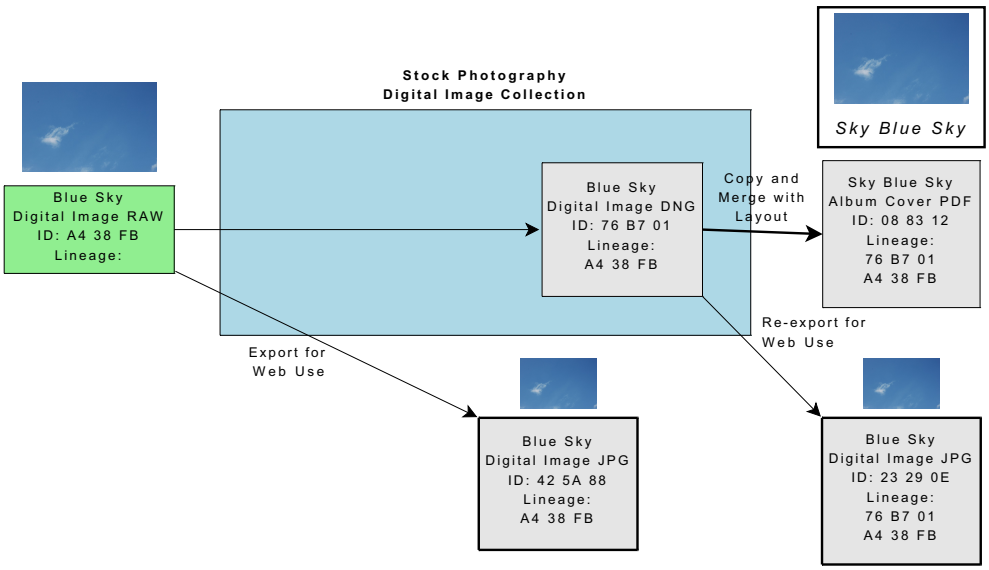


Figure 3.4: Ancestry graph example for ancestry flattening (flattened digital object A8 76 30)

The flattening function $flatten(A, do)$ converts an ancestry graph A to a flattened ancestry graph A', removing the digital object do ($flatten(A, do) \rightarrow A'$). To retain the structure, the edges of the removed digital object need to be re-orientated. Since all ancestors of the removed digital object are by definition also ancestors of the descendants of the removed digital object, the ancestry relation needs to be re-assigned to all descendants of the removed digital object. Therefore, the $init$ and ter maps of the ancestry graph need to be updated to reflect this. First, the set of ancestors AN of the digital object do are identified by $init(E(do))$, where $E(do)$ is the set of all edges at the vertex do ($AN := init(E(do))$). Second, the set of descendants DE of the digital object do are identified $ter(E(do))$ ($DE := ter(E(do))$). The flattened ancestor graph A' is a pair (V', E') of disjoint sets of vertices V' and edges E', where $V' = V \backslash \{do\}$ and $E' = (E \backslash E(do)) \cup E_n$, where $E_n := E(AN, DE)$ is the set of all edges from all ancestors to all descendants. Therefore $init(E'(DE)) \supseteq AN$ and $ter(E'(AN)) \supseteq DE$.

Ancestry flattening can be reiterated for all digital objects which need to be removed from the ancestry graph. This may also include primal or terminal digital objects. Since these special cases have either no ancestors or descendants, they and their edges can be removed without the need for new edges. Figure 3.4 shows the resulting ancestry graph after flattening the digital object *A8 76 30* from the initial graph (figure 3.2).

Ancestry Refinement Ancestry refinement allows the addition of detail to an ancestry graph by adding additional ancestry relations and digital objects to an ancestry graph. Ancestry refinement is used to update the ancestry of a digital object to reflect finer grained information about the ancestry of a digital object. It also reverses the effects of ancestry flattening. So, a digital object

of which the ancestry has been flattened can be updated with the information removed using the aforementioned procedure. An example of ancestry refinement in use is the re-introduction of a previously flattened digital object into the ancestry environment. The previously removed ancestry information can be re-introduced into the digital object so that it contains all of the ancestry information. Ancestry refinement may also update the ancestry of a digital object to reflect added information about the ancestry of a digital object. If more information on the intermediate digital objects is available, this can be incorporated into the ancestry graph. We will not investigate ancestry refinement any further, but it is an interesting topic for future research.

3.3 Advantages and Limitations

In this section, we discuss the advantages and limitations of our theoretical approach towards identification and ancestry of digital objects.

We have provided a precise definition of equality between digital objects on the bit level. This definition ensures that only digital objects with exactly the same content are considered equal and makes the determination of equality trivial and mechanical. Implicitly, all context outside of the digital object is ignored, thereby reducing complexity and eliminating any external dependencies from the equality considerations. However, the restriction to the bit level does not allow a higher level comparison between digital objects, for example between two versions of a PDF file that display identically, one with embedded fonts and the other one without. The files are different, but to a user, the rendered result is exactly the same. Additionally, two digital objects that reference another digital object will be rendered differently if the referenced digital objects are not equal, even if the referencing digital objects are equal according to our definition. An HTML document, for example, will be rendered differently if a referenced style definition is changed, even if the document itself is unchanged. Generally, digital objects will only be rendered to the same result, if all inputs to the rendering process are equal (according to our definition of equality), assuming a deterministic process.

We have also provided the theoretical basis for the identification of digital objects relying exclusively on their content, in accordance with our definition of equality. On this basis, digital objects can be identified independently of the information systems which manage them. We define *identification functions*, which provide identifiers using the digital object as input. Thereby, the digital object attains an identity which is independent of any external system. Additionally, we propose the use of well-researched collision resistant cryptographic hash functions as identifier functions for digital objects, when the determination of uniqueness and integrity is a requirement. However, since in our approach the identification of a digital object is limited to the digital object, additional information on the digital object, however useful it might be, will not be taken into account when defining an identifier. Therefore, information which is not embedded in the digital object will not be part of the identification. For example, an identical digital love letter could be

identified and interpreted differently depending on whether it was found in the inbox of a man's wife or his lover. The location of an object is always a part of its identity in the real world. This can only be duplicated in our approach if the location information forms part of the digital object.

We have assessed the possibilities of incorporating identifiers (i.e., the result of identification functions) in existing identifier systems. We conclude that a part of the identifiers defined by the identifier systems must be able to be chosen freely for the output of an identification function. The identification function can be defined in conformance with the identifier system's identifiers. The possibility of choosing between different existing and proven identifier systems for the identification of digital objects according to yet unknown requirements, adds to the flexibility of our approach. However, since no new identifier system is proposed, the weaknesses and strengths of existing identifier systems have to be accepted. The design of a new identifier system for digital objects would have better suited the needs of digital object identification.

Finally, we defined digital object ancestry, its representation and basic manipulations on the ancestry. In our approach, we concentrate on the digital object and its ancestors, which are also digital objects. The ancestor-descendant relations between digital objects are based on the identification defined beforehand. However, the processing involved in creating a digital object is not taken into account; only digital objects used in the creation of a digital object, or their ancestors, are included in the ancestry, for complexity and volume reasons. This limits the usefulness of our approach when retracing the processing history of digital data. Moreover, all non-digital object inputs (policies, human interaction, etc) are not included in the ancestry.

3.4 Summary

We have defined the theoretical basis for context-free equality and identification of digital objects, thereby making equality and identification independent of information systems. We have defined the requirements for the embedding of identifiers into existing identification systems. Additionally, we have defined the representation of ancestry as a graph and defined basic manipulation functions on the graph.

In the next chapter, we present the requirements for the integration of metadata with digital objects. The method called SIMPLE implements the requirements defined. We will define a generic, extensible data model which represents digital object metadata and its implementation to the bit level. Additionally, we will show the results of a case study, applying SIMPLE in the State Archives of Appenzell Ausserrhoden.

4

Lineage Integration

In this chapter, we show the integration of lineage with arbitrary digital objects. We speak of "lineage" integration instead of "ancestry" integration since the focus of the integration is a specific digital object and its relation to its ancestry. As shown in section 2.6, metadata can be bound to digital object in different ways. We consider metadata integration to generally be more useful than lineage integration, without posing additional difficulties. We will therefore first lay down the requirements for and the possibilities for metadata integration with different digital object formats before elaborating on our approach of integrating metadata with digital objects. As lineage metadata is part of the general metadata the integration of metadata includes and forgoes the integration of lineage metadata.

4.1 Requirements

Here we define and motivate the requirements of metadata integration with the digital object. The requirements mainly concern the digital object; the information systems managing the digital object and the institution operating the information system only affect the requirements where mentioned.

4.1.1 Identify Digital Objects Independent of Information Systems

The identification of digital objects is often coupled to the information system managing the digital object, with its identifier(s) being defined by the information system (section 3.1). An identical copy of a digital object managed by another information system will be identified differently governed by the rules of the other information system. Additionally, if a digital object can only be identified within an information system and not as an independent entity the digital object will only be accessible as long as the information system is active. The requirement is therefore that *digital objects must be identifiable independent of any information system*.

4.1.2 Inhibit Digital Object – Metadata Dissociation

Metadata puts the digital object into context. A digital object without metadata is of less value than the same digital object with metadata. Because of the broadened context provided by metadata, the digital object is more self-sufficient. This is a two-way relationship: the metadata helps to interpret the digital object and the digital object helps interpret the metadata, since there are intrinsic properties of the digital object that may not be present in the metadata. The digital object puts the metadata into a context. If the metadata of a digital object is altered in a way that makes it impossible to find the associated digital object then the metadata cannot be supported by the digital object and therefore loses most of its evidentiary value. The evidentiary value of all instances of metadata which refer to the digital object is negatively affected if the digital object they refer to is no longer accessible. Any assertion made by a referrer to the digital object loses significance if the assertion can no longer be verified by following the link provided by the referrer. For example, a citation or reference which cannot be backed up by its source is worthless. The requirement is therefore that *the digital object and its metadata must be inseparable*. This requirement backs the requirement that the identification of digital objects must be independent of any information system (subsection 4.1.1); if the dependence of the information system on the association between the metadata and the digital object is eliminated, the association will be less fragile.

4.1.3 Simplify Metadata Identification and Access

Using the digital object as a starting point, the associated metadata can usually not be identified, as a digital object cannot reference its metadata. Even if referencing were possible, a digital object cannot be aware of all of its metadata. An electronic article, for example, cannot be aware of all publications that reference the article or the metadata held in scientific literature digital libraries and search engines such as [CitesSeer, 2008]. Additionally, the metadata inherent to the digital object is usually stored format-dependently according to the format of the digital object. The metadata is therefore only accessible if the digital object format and its specification is known. However, metadata will only be used if its access is straightforward, therefore the *metadata must be easily identifiable and usable*. This requirement backs the requirement of the identification of digital objects to be independent of any information system (subsection 4.1.1), since the link between the metadata and its digital object will no longer depend on an information system.

4.1.4 Secure Integrity

Integrity ensures that changes to the digital object can be detected. Securing integrity means protecting the digital object and its metadata against changes (malicious, intentional, or unintentional) that cannot be retraced. In most current implementations of information systems, changes to the digital object cannot be detected. Even in an information system that supports the detection

of changes, for example through the use of checksums or hash values, metadata changes cannot be detected and traced as easily. Even small changes in the metadata, like changing the checksum value or the storage location (URI or URL) of the digital object, will diminish the evidentiary value of the digital object and its metadata. Because the digital object cannot be found, its integrity will be questioned, etc. The requirement is therefore that *changes to a digital object or its metadata must be detectable*.

4.1.5 Make Lineage Relationships Explicit

The lineage of a digital object identifies the digital objects which lead to the creation of that digital object. These digital object ancestors provide additional context to the digital object and may aid in answering questions about the digital object. Additionally, it should be clear which digital object is currently being used, whether or not ancestors exist, and how the ancestors can be identified. The requirement is therefore that *the lineage of every digital object must be included*. This requirement backs the requirement of the identification of digital objects independent of any information system (subsection 4.1.1) since digital objects need to be identified independent of any information system if the lineage spans more than one information system.

4.1.6 Accommodate Arbitrary Metadata

The metadata scheme must allow for metadata reuse since the creation of metadata is costly, time-consuming, and should not have to be repeated. Furthermore, it must be possible to accommodate new and as yet undefined metadata. The requirement is therefore that *it must be possible to accommodate arbitrary metadata*.

4.1.7 Invariant Towards Software and Technology

Information technology is a synonym for change. New software is created constantly, replacing old software; new technologies emerge superseding or supplementing older technologies. In the long term, metadata integration must be *independent of any specific software or technology* since any of them may become obsolete. This requirement implies that the metadata integration must be implementable with various software and technologies.

4.1.8 Minimize Complexity

Every layer of complexity adds possible points of failure. An example is the use of encryption where the content of a digital object remains unintelligible if either the decryption key or the decryption algorithm is unavailable. Any form of encoding is a form of encryption while decoding

is the equivalent to decryption (although without the need for keys). The requirement is therefore that the method must be *as simple as possible*. This is commonly referred to as KISS principle [Bennett, 1997].

4.2 The SIMPLE Method

In this section, we present the *SIMPLE* method for integrating metadata with digital objects. SIMPLE is a recursive acronym that stands for "*S*imple *I*dentifiable *M*etadata with *P*ersistent *L*ineage *E*mbedding". The method with this name is composed of a mandatory part (SIM), which defines the identification of metadata and an optional part (PLE), which defines the embedding of lineage into digital objects. This section will give an insight into the elements of the method.

In the SIMPLE method, the metadata becomes part of the digital object; a stark contrast to the digital object and metadata management methods we know of to date. The digital object and part of its metadata form a new digital object from which the metadata cannot be separated without compromising the integrity of the digital object as a whole. Additionally, all changes including the joining of the digital object and its metadata are documented and can partly be undone using the information provided by the digital object, or, more specifically, the metadata part of the digital object. Every change, be it in the digital object or its metadata, is reflected in the metadata and will therefore create a new digital object. It should be possible to use the resulting digital object in its native environment as if no additional metadata were present. The metadata part of the digital object may not be present in a manipulated successor, especially if the environment manipulating the digital object does not support the SIMPLE method. For example if a TIFF image with SIMPLE metadata is edited by an image processing software which is unaware of the metadata, it will either be discarded or it will no longer be accurate for a changed version of the digital object, since the digital object described is no longer the digital object the SIMPLE metadata was originally bound to. The adoption of the original SIMPLE metadata will provide clues to the digital object ancestor used in the creation of the new digital object. We speak of the SIMPLE digital object (SDO) when SIMPLE metadata is integrated with a digital object.

4.2.1 Simple

Simplicity is the first of two elements of the mandatory SIM-part of the SIMPLE method. For a method to be accepted in, but not confined to, the context of archives, the method needs to be simple and durable to reflect a lowest common denominator, and its implementation should lead to an increased productivity and stability. These needs are hard to measure and are interdependent. An (overall) simpler and therefore more durable method will lead to stability [Lorie, 2001], while the lowest common denominator will normally be simple (although not necessarily durable).

Since an archives must guarantee the preservation of records in its custody for a potentially unlimited amount of time, the original producers will not be able or willing to compensate the archives for the preservation. The SIMPLE method is so simple that any person with a minimal IT background is able to understand and supervise its implementation, a requirement shared with [Quenault, 2004]. To achieve this we use the following complexity reduction principles:

- Principle of Least Astonishment: The definitions, technologies, models, and schemes should be defined in a way that least confuses the user, that is, in a way that is consistent with the user's experiences and his mental model; it should be intuitively understood. This principle is commonly used in software development, see for example [Vermeulen et al., 1999, p. 1], [Raymond, 2003, p. 20], and [Bara, 1995, p. 338].

  When looking at the SIMPLE method, it should be so intuitive that anyone familiar with the definitions, technologies, models, or schemes used to implement the method is able to understand its use.

- Natural Language Descriptions: Natural language has been the basis for the preservation of knowledge over time. [Lorie, 2001] proposes the use of natural language as a bootstrap-process for a digital library. The SIMPLE metadata is therefore enhanced with additional natural language texts that help humans understand the purpose and meaning of the metadata structure and content. This makes the information explicit at the location where it is needed.

  Natural language text is not easily understood by computers, but for thousands of years it has been the main form of non-oral knowledge. The variations of text in a natural language are endless and there is always room for interpretation, but within the context it provides helpful clues on how the SIMPLE metadata is meant to be used.

Figure 4.1 shows two examples of metadata values which are hard to make sense of without additional information or description. In contrast, figure 4.2 is much clearer for a human to interpret since a short statement clarifies the semantics of the metadata values. The statement is actually meta-metadata that is present in the metadata definition. The decision on how much description metadata is needed to attain a defined goal depends on its practical use. Metadata for digital objects used for archival purposes will need a more detailed description than for digital objects which are exchanged between automated systems.

4.2.2 Identifiable Metadata

Metadata must be able to be unambiguously identified as metadata. The lowest common denominator for the definition of metadata is "Metadata is data about data", so metadata is dependent on other data. We use the term metadata in the sense that data describing data is metadata. This does not preclude that the describing data cannot be used as data in another context. Using the

In XML:

```
<contact>031 371 31 70</contact>
```

In EXIF:

```
XResolution: 300
```

Figure 4.1: Metadata values without additional description

In XML:

```
<!-- Telephone number of the person responsible for
     running the digital archives -->
<contact>031 371 31 70</contact>
```

In EXIF:

```
\# The number of pixels per ResolutionUnit in the
\# ImageWidth direction.

XResolution: 300
```

Figure 4.2: Metadata values with additional description

aforementioned simplicity principles we label the metadata as 'metadata', therefore making it easy to identify it as metadata.

Byte Level The start and end of the metadata must be clear, therefore it must be structured in a way that allows the identification of start and end of the metadata on an byte level. To satisfy these requirements we use XML [Bray et al., 2004] in the UTF-8 [Yergeau, 1998] encoding as representation for the metadata. The use of XML emphasizes the human readability of its contents [Heuscher, 2003].

To allow the identification of metadata within a digital object of an unknown format, we propose the identification on the basis of a unique byte sequence. We choose the byte sequence *<metadata describes=* as identification (magic number) for the start of the metadata (the less than and equal signs are part of the identifying sequence). In Unicode [Allen et al., 2007] the magic number is "LESS-THAN SIGN" code U+003C followed by the codes for the characters of 'metadata describes' where the white space is a "SPACE" code U+0020 and ending with an "EQUALS SIGN" code U+003D. We chose to use the XML attribute *describes* because metadata always describes data. The content of the describes attribute is not defined. However, we advise the use of a reference to the described entity. If the described entity is the digital object and the metadata is a part thereof, the attribute is to be left empty as this implies a reference to the digital object itself [Berners-Lee et al., 2005]. The byte sequence *<metadata describes=* is unique enough to allow

the identification of the metadata. A web search on Google yielded no result for XML documents with this character sequence on 5. May 2006 and 18. June 2008. Since search engines provide their services on word (not byte) basis we used the UTF-8 encoding to transform the byte sequence to a character sequence and searched for the sequence in XML files. The link used was `http://www.google.com/search?q=%22metadata+describes%22+filetype%3Axml`. The search for the $<$ and the $=$ sign was conducted manually. The other major search engines (Yahoo and MSN Search) did not provide the possibility to narrow down the search results by an arbitrary document type. They could therefore not be used for this research. The end of the metadata is defined by the XML standard constraints, therefore ending with the byte sequence *</metadata>*. We call the defined XML document *SIM XML*.

There are two additional properties that the SIM XML must adhere to. First, the SIM XML should not include any new line characters since this impedes the accessibility of the original digital object if the file format definition is line based like file formats of different programming languages and character separated values formats. This requirement can easily be met by replacing the new line characters with their entity reference. The new line character is represented by "*
*" in XML. Second, the SIM XML must not include any XML comments as they would impede the ability to embed SIM XML into XML comments, since XML does not support comments within comments.

Metadata Design The XML element *metadata* is the root element of the XML metadata document. Besides the name of the root element, that of its first attribute (*describes*), and the encoding of the metadata, we do not put any constraint on the content of the XML document (for the SIM part). This is intentional, since it allows the users to define their own metadata adapted to their environment. There is no design at this level apart from the definition of the root element and one attribute name.

Metadata Re-Use In practice, metadata is valuable because its creation is mostly very labor intensive and therefore expensive. Metadata comes in various forms, ranging from paper indexes over microfilm to various digital forms like Encoded Archival Description (EAD) [LOC, 2002], The modular structure of XML allows any XML documents to be embedded into a SIM XML document. Therefore any XML metadata document can directly be added. Other text-based metadata without control characters can be embedded after escaping two reserved characters. The XML standard version 1.1 disallows the Unicode code of 0. It defines "*&*" (ampersand, Unicode code 38) and "$<$" (less than, Unicode code 74) as reserved characters which have to be escaped to "*&*" and "*<*", respectively. Text-based metadata with control characters must either escape the non-XML characters to conform the XML specification or it must be treated as binary metadata. Binary metadata can either be mapped to XML and represented as an XML document or it can be encoded in $Base_{64}$ [Josefsson, 2006], the standard encoding for binary data in XML, before it can be inserted.

The reused metadata can be extracted from the SIM metadata either by means of a stylesheet, by deleting all elements that are not part of the embedded metadata and un-escaping any escaped characters for text-based metadata, or by decoding the $Base_{64}$ encoded metadata for binary metadata.

4.2.3 Embedding

In this subsection, we lay out the options of embedding SIM XML metadata in arbitrary digital objects. We will subdivide digital object formats taking into account their ability to include SIM XML metadata.

Characterizing Digital Object Formats Digital object formats, commonly called file formats, define how information is encoded in the digital object. There are two basic types of data organization within a digital object. The first type defines an internal format that allows the dimensions of the digital object (its size) to be deducted from the contents of the digital object itself. We call this type of digital object format *confined*. Applications rendering or manipulating this type of digital objects will not access data outside of the internally defined dimensions. Therefore, data outside of these dimensions will be ignored. Examples of confined digital object formats include binary formats like ZIP, DOC, TIFF, and JPEG. The second type of data organization defines the format so that the dimensions cannot be deducted from the contents of the digital object. We call this type of digital object format *open-ended*. The dimensions of the digital object are imposed externally for example by a file system or database management system. Examples for open-ended formats include text formats like XML, CSV, or TXT. Contrary to confined formats, applications rendering or manipulating open-ended formats cannot and will not ignore any data within the digital object. Therefore, if metadata is to be added it must be added in accordance with the format. One communality found in almost all digital object formats is the option to embed comments. Comments are normally ignored by an application processing digital objects since their main objective is to add context or information in unstructured form for humans. Most comments are in plain-text. Before a comment can be added, the format of the digital object must be known, as there is no universal agreement on the form of comments. The addition of a comment changes the content and structure of the digital object. Therefore, the application adding the comment must be able produce a digital object in accordance with the format specification. Some formats, like TXT or CSV, do not define comments. For the maximum size of the embedded comment metadata of selected file formats refer to Table 4.1.

Embedding Metadata in Digital Objects Metadata is normally held outside of the digital object it describes, in the information system managing the digital object or other information systems. The management of metadata outside of the digital object simplifies the creation and modification of the metadata, as the digital object might undergo multiple changes, which may need to be

Format	Approx. Max. Comment Size	Remarks
ZIP	32 KB	The addition of a comment field changes the structure of the digital object.
JFIF v1.02	$x*64$ KB	The addition of a comment changes the structure of the digital object. A comment is defined by a JFIF marker `ff ee` followed by the length (2 bytes, in bytes) of the following null-terminated comment string (including the length bytes and the null-byte themselves).
EXIF Image v2.2	$x*64$ KB	The addition of a comment field changes the structure of the EXIF data.
TIFF v6.0	4 GB	The addition of a comment field changes the structure of the digital object.
EXIF Audio v2.2	$x*4$ GB	The addition of a comment field changes the structure of the EXIF data.
WAV	$x*4$ GB	The addition of a comment field changes the structure of the digital object.
PDF	unlimited	
TXT	unlimited	The comment functionality depends on the type of text format.
XML	unlimited	XML documents may allow the insertion of the metadata as an organic part of the original XML document (depending on the XML format definition).

Table 4.1: Maximum size of comment fields of selected digital object formats

reflected in the metadata. As described in section 2.6 this solution is not suitable for use with digital objects independent of information systems because the connection between the digital object and its metadata is only guaranteed by the operational information system managing the digital object or its metadata; information system independence can only be achieved if the metadata is packaged with the digital object. We focus on the embedding of SIM XML metadata as defined in the previous subsections 4.2.1 and 4.2.2. The metadata can be integrated in the digital object according to its format.

For confined formats, the metadata can be embedded using *binary append*, thereby retaining the original structure of the digital object. This procedure can easily be undone since the original data and the metadata can be distinguished quite simply. Alternatively, the metadata can be embedded in a comment structure of the digital object using *format specific embedding*, if comments are supported by the format. This procedure disrupts the original structure of the digital object. However, in the resulting digital object the integration of the metadata is superior to the integration outside of the structure imposed by the format, since the metadata can be accessed with tools supporting the format. Additionally, the fusion of data and metadata is more profound, since this metadata will possibly remain in future, modified versions of the digital object, contrary to metadata embedded outside of the format specifications. However, if metadata is embedded as a comment, this metadata can only be removed by deleting the comment, thereby changing the structure of the digital object. Both forms of packaging are further described in subsection 2.6.2.

For open-ended formats with a comment functionality, the metadata can be embedded using comments. Depending on the format definition of the digital object, the comment containing the metadata can be embedded at different locations inside the digital object. The options on how to embed the metadata are similar to those for confined formats. It can be embedded at defined places within the digital object, thereby achieving a stronger integration and a change in the structure of the digital object, or added as a comment at the end of the digital object, retaining the structure of the digital object. A special case of metadata integration is the integration of metadata with digital objects of XML formats. Since XML is only a meta language for the definition of formats the possibility of natively integrating SIM XML with a digital object of XML format depends on the format definition of the XML format; its structural definition. If the structural definition of the digital object in XML format allows the integration of arbitrary XML data, SIM XML can be integrated at one of the structurally defined locations.

For open-ended formats without a comment functionality, the integration of metadata cannot be done without consequences for the later processing of the data. Therefore, the process of embedding needs to be undone before the digital object can be processed further. However, for data that is merely intended for human interpretation, the undo step might be skipped if the metadata can clearly be distinguished from the original digital object. Both the automatic undoing and the clear demarcation of metadata suggest that metadata integration with open-ended formats without a comment functionality should be restricted to the adding of the metadata at the end of the digital object.

Depending on the goals of the metadata integration, the format of the digital object, and the point in time when the metadata is integrated, different strategies can be applied. If modification of the original digital object is to be kept to a minimum (in digital archives for example) then the metadata should be appended at the end of the digital object, possibly as a comment, depending on the type of the digital object. However, if a new version of the digital object can be created, and the adherence of the metadata to the digital object has a higher priority than the integrity of the original digital object, then the metadata should be embedded within the digital object as a comment.

The SIMPLE method does not differentiate between metadata that was appended or added in the form of comments. The format specification of SIM XML defined in subsection 4.2.2 can be retrieved from any digital object by analyzing (parsing) its content. For the extraction of the SIM XML, metadata the format of the digital object need not be known, since the metadata can be found in and retrieved from any digital object, irrespective of its format.

Establishing Digital Object Integrity The verifiable integrity of a digital object is established through a process we call *sealing*. This process uses only the information provided by the digital object itself; there are no external dependencies.

Every change in the bit stream of the digital object leads to a change in the hash value of the digital object. Therefore, the hash value of the digital object cannot be included in the digital object itself without special precautions. XML-Signature [Eastlake et al., 2002], for example, removes the signature part of the XML document for the signing and verification process and the result of the signing process is added later to the XML document. To circumvent this dependency, we use a placeholder, which will be replaced with a calculated hash value of the digital object. The placeholder must have the same length as the hash value, since the size of the digital object is often used as an additional indicator for its integrity.

The sealing process is divided into two steps. First, the hash values per hash function are calculated from an unsealed SIMPLE digital object (SDO). The unsealed SDO contains pre-allocated locations to insert the resulting hash values, the placeholders or "dummy values." In the example the placeholder is a string of full stops (.) of the same length as a $Base_{16}$ encoded md5 hash value. The XML tag <md5> indicates that the hash function is md5:

```
<md5>................................</md5>
```

Second, the hash values are calculated. Naturally, the placeholder, as part of the unsealed SDO, contributes to the resulting hash value. Then the placeholders are replaced with the calculated hash values:

```
<md5>10a8f787016eb0fa0fba6a6ccf119756</md5>
```

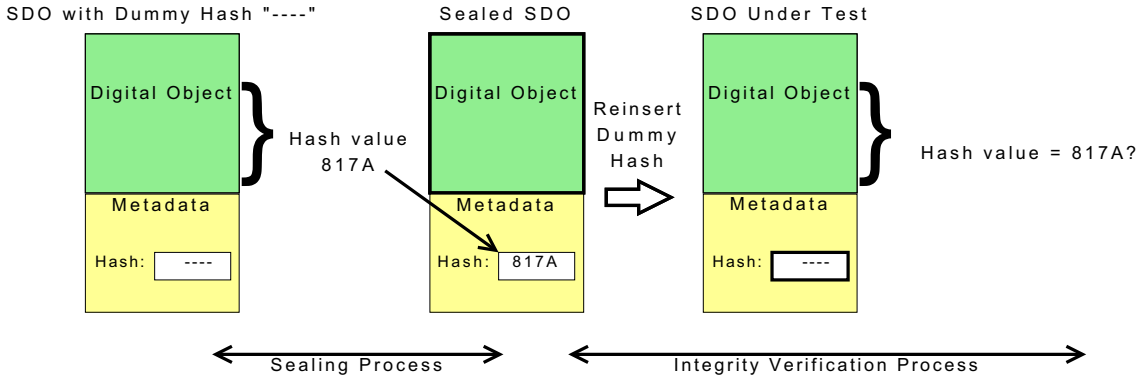


Figure 4.3: Sealing and verifying a SIMPLE digital object (SDO)

The hash values calculated from the sealed SDO will differ from those stored in the SDO after the insertion of the hash values into the unsealed SDO; the sealing process changed the SDO.

After sealing a digital object, any change to the digital object and its SIMPLE metadata can be detected. The sealing does not protect against malicious manipulation of the digital object by a skilled attacker. He would insert a matching hash value into the digital object after altering the content of the digital object, thereby re-sealing it. This new hash value would, however, be different than the original hash value. Additionally, the identification of the digital object changes as well. It is possible to use cryptographic techniques to sign or encrypt the digital object. While the signing of a digital object does not alter its contents, its encryption would introduce at least one additional layer of complexity. It couples access to the digital object with the availability of the cryptographic infrastructure, including cryptographic certificates, public keys, and certification authorities. We therefore do not encourage the use of cryptographic techniques to encrypt the digital object.

Verifying Digital Object Integrity To verify the integrity of the digital object, the sealing process is reversed. First, the hash values are replaced with their placeholders. Then the hash values of the resulting (unsealed) SDO are compared to the hash values stored in the sealed SDO. If all value pairs per hash algorithm are equal, then the integrity of the SDO is verified. The process of sealing and verifying the integrity of a SDO is depicted in figure 4.3.

The lineage information presented in the following subsection 4.2.4 can be used to aid in the verification of integrity. The lineage information includes the identification of the ancestors of the digital object, and if a source of the ancestors can be found then the integrity can be checked at the source of the digital object. An alternative approach is to search the Internet or specialized (not yet existing) services for the hash value of the digital object and any ancestor object. The results of the search can then be cross-checked against the digital object, thereby revealing all discrepancies in either the digital object or the lineage.

4.2.4 Persistent Linage

Lineage (also called pedigree, provenance, audit trail, or chain of custody) describes the ancestry of the digital object. This ancestry includes the sources (ancestors) that were used in the creation of the digital object as described in section 3.2. For a reliable ancestry, every digital object must be uniquely identifiable. Therefore, we will first concentrate on the identification of digital objects after which the integration of lineage with the digital object will be defined.

Identifying Digital Objects A digital object is normally identified by an identifier, which is managed by an information system. This identifier may change without the digital object being altered. The digital object is identified by its identifier and the information system managing the identifier like `c:\boot.ini` on the harddisk of a computer or the 4th `img.jpg` file in an email message. The cornerstone of our lineage definition is the unambiguous identification of every digital object in any information system. Assuming that the information systems are unaware of this, the identification must either be kept inside the digital object (analog to the reference code in a document) or it must be a unique intrinsic property of the digital object. Section 3.1 describes this in more depth and as defined therein we will use the output of cryptographic hash functions applied to the digital object as identification for digital object.

Dealing with Hash Collisions The probability that two different digital objects will produce the same output if a cryptographic hash function is applied is very small. Notwithstanding, the probability is not zero. The probability of collisions can further be diminished by using more than one cryptographic hash function. In the extremely unlikely event that the values of all cryptographic hash functions collide, the digital object could be changed in a transparent manner. This can be done, for example, if the digital object contains SIM XML, by adding an XML white space character after the *identifies* attribute of the metadata tag, so that results of the cryptographic hash functions no longer collide. However, collisions can only be avoided if they are detected. The prospects for the detection of a collision depend on the scope of the applied search for detection. If the detection scope is limited to the digital object, then the detection can only encompass the digital object itself. This scope is useless unless the digital object contains metadata about its ancestry. In this case the current identifier of the digital object can be tested against the identifiers of the digital object's ancestors. Broadening the scope of an application checking for collisions could allow it to check for collisions within the context of the information system managing the digital object. An even broader verification could also take place involving specialized services, which do not exist at this time.

However, for the most part, the choice of one or more cryptographic hash functions should bring the probability of hash collisions down to an acceptable level. The decision on which hash function or combination of hash functions to use and the effort put into the search for hash collisions can be defined according to the needs of the implementation.

Establishing Ancestry Before the lineage of a digital object can be added to the digital object as metadata, it needs to be collected. We will build upon the ancestry graph described in section 2.5. The lineage of a single digital object can be established in one of the following approaches:

Centralized The information systems managing the digital object collect the identifiers of all digital objects used to create a digital object. The information system provides ancestry data on the digital object upon request.

Decentralized The digital object itself contains its lineage or parts thereof. It must therefore at least contain references to its ancestors.

The centralized approach is simpler to manage since all ancestry information is available centrally and the lineage of a digital object within the context of an information system can easily be established. However, the context of the information system may be too limiting for most applications of ancestry, since digital objects can be shared between information systems by copying a digital object from one information system to another. If the lineage is to be preserved, any digital object taken out of the context of an information system with centralized ancestry management and acquisition must be augmented with its lineage as additional metadata.

The decentralized approach relies on the digital object to contain its lineage and every processing step added to that lineage. Here, the digital object must at least contain a reference to its ancestors. This reference is the starting point for the creation of the ancestry of the digital object. Further information on the previous ancestors can then be found in the ancestors. This recursive technique can be applied until no more references to ancestors are found. It is not necessary that all intermediate digital objects are preserved if the lineage within the digital object spans more than one generation or one processing step. However, if a complete ancestry is to be achieved, at least the primal digital objects referenced in the lineage of the digital object must exist and, again, contain their lineage.

The example in figure 4.4 shows the ancestry graph of a digital still image. After the creation of the original still image, it is ingested into the stock photography digital image repository. In the course of the ingest process the original image is transformed into a TIFF file (ID A8 76 30), which is then stored in the image store. The digital object is converted for web use (ID 42 5A 88) and is published on the Internet. In the course of time, additional metadata on the still image is generated which is added to the image when the digital objects of the image store are migrated into the new DNG format. Thereby, a new digital object (ID 76 B7 01) is created. This digital object is then used for the album cover of the 'Sky Blue Sky' album which is represented by a PDF file (ID 08 83 12). Again the digital object is exported for web use (ID 23 29 0E) and published on the Internet. Here the primal digital object is the digital still image (ID A4 38 FB).

Since the organization distributing the image has an interest that it is known as the source of the exported versions of the digital object, we believe that it will want to provide easily accessible metadata to identify at least one of the ancestors of the exported digital object.

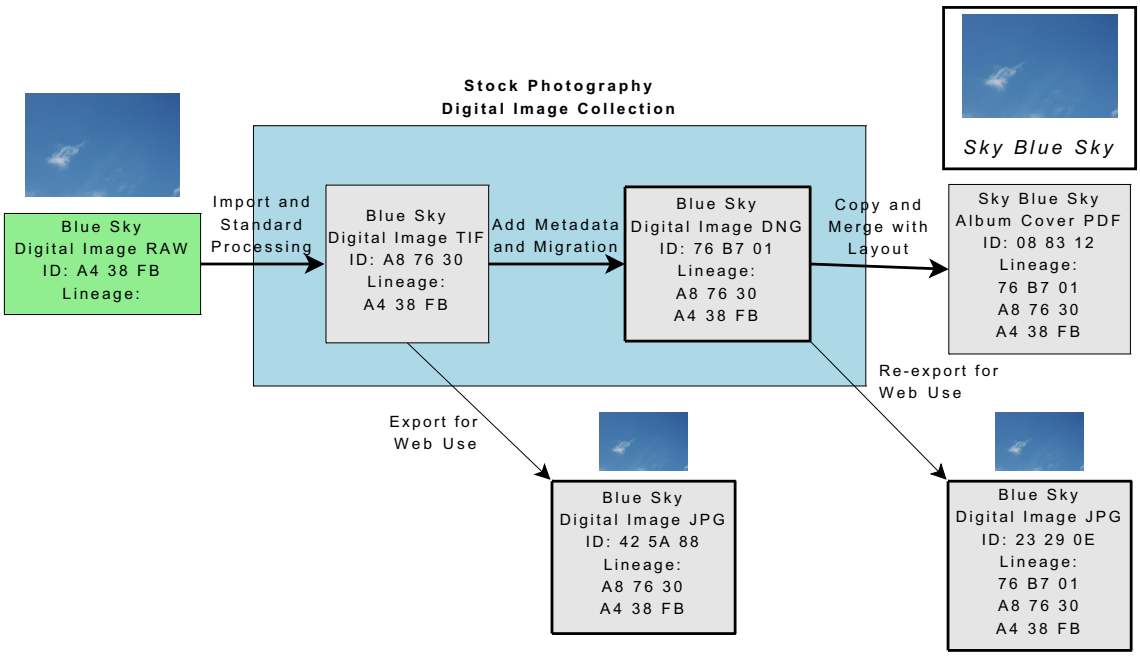


Figure 4.4: Example lineage graph

Embedding Lineage Lineage metadata is a specialized form of metadata the embedding of which has been described in subsection 4.2.3. Therefore, a suitable method for the mapping of a lineage graph to XML needs to be defined. We will propose a method to be used in the prototype in chapter 5. A well-founded definition of mapping would go beyond the scope of this thesis. However, we foresee the use of domain specific mapping like the XMP media management scheme for the use by digital asset management (DAM) systems [Adobe, 2005].

4.3 Advantages and Limitations

In this section, we discuss the advantages and limitations of lineage integration with arbitrary digital objects, its requirements, and of the introduced SIMPLE method.

4.3.1 Storage

Using our method, any metadata can be integrated with the digital object. We specifically do not restrict nor require metadata to be present, with the exception of lineage and integrity metadata, which are optional but specified nevertheless. Since there are no restrictions on the metadata for certain digital objects and with time the metadata will use more storage space than the actual data. Additionally, digital objects with metadata use more resources than digital objects without

metadata, be it in storage and transmission or in the processing of the digital object. The verbosity of XML adds to the storage requirements. And, since SIMPLE is based on metadata in digital objects, metadata may need to be duplicated in every digital object concerned.

4.3.2 Flexibility

We define a container for XML metadata which can be extracted from any digital object, irrespective of its format. The content of the container can be filled with whatever metadata is deemed necessary. This makes it possible for anyone using the SIMPLE method to use the metadata best suited for their needs, be it the inclusion of all the metadata stored within digital archives with the aim of rebuilding the archives metadata after a catastrophic incident, or just the inclusion of contact or copyright data for a later user to know where the digital object originated from. However, the flexibility puts a burden on anyone using the SIMPLE method. SIMPLE does not offer a predefined plug-in package that can be used without the involvement of stakeholders. Devoid of a defined format, SIMPLE in its current form can only be used for reference and human interpretation. Automated processing of metadata is only possible if its format is defined. Specific metadata is defined for integrity and lineage. However, this metadata is not defined in any form of XML structure specification.

We add the defined container to digital objects in a form dependent on the format of the digital object and with the aim of metadata integration. This counteracts the premise that the same mechanism should be used for all digital objects independent of their format. However, some digital object formats are not amicable towards the addition of metadata. For these formats, a specific handling has to be put into place to allow the unhindered processing of these formats, which adds to the complexity overhead imposed by SIMPLE. In contrast, the detection of SIMPLE metadata is straightforward and any informatics student could extract all metadata of all digital objects on a harddisk within a couple of days.

4.3.3 Identification

In our method, identification of a digital object relies exclusively on the content of the digital object. This allows identification to be independent of any information system, thereby strengthening the autonomy of the digital object. However, this view differs from the common view where the “residence” of the digital object adds greatly to its identity. Also, with our definition of identity, small changes to a digital object, which may be of no importance to the user, like the adding of space character at the end of a text file, lead to a different digital object. This fragility of the identification does not represent a common mindset where digital objects are perceived as highly malleable and changes to the digital object are not reflected in its identity.

4.3.4 Lineage

In the portrayed SIMPLE method, the focus for the ancestry of a digital object lies in its lineage, that is, in its ancestors and their ancestors. The broader scope of the complete ancestry of a digital object is masked and relationships not directly linked to the digital object are suppressed. This leads to a plainer, less complex view on the ancestry of the digital object. Additionally, the maintenance of lineage for an isolated digital object can be done without the support of an information system by adding the lineage of all digital objects involved in the creation of the digital object to the resulting digital object. However, in order to put the digital object into a broader context the complete known ancestry may be of interest.

4.3.5 Integrity

With our method, the integrity of a digital object covers its content and the metadata associated. Integrity is secured by exploiting the collision resistance of cryptographic hash functions using the digital object data and metadata as input for both integrity and identification for the digital object. We describe a technique (sealing) for integrating the hash values with the metadata that allows us to check if any changes have occurred in either the data or the metadata. However, the integration of the hash values is not a defense against malicious attacks on a single digital object since the attacker can use the same sealing technique to make the digital object appear unaltered. This attack could be prevented by the use of cryptographic signatures, the use of which we discourage due to their complexity. However, cryptographic signatures gain more and more relevance in the market and their acceptance is backed by the European Commission.

4.3.6 Adoption

Our method allows the integration of arbitrary metadata on the level of digital objects while preserving their format. Therefore, the digital objects can be processed within their customary environment regardless whether or not metadata is present in the digital object. This lowers the barrier for an adoption of this method since no changes in the normal processing of the digital object is needed. However, because no change is needed, the update of information systems to enable lineage information collection might not take place, because it “works well without.” But, if the information system or applications manipulating digital objects do not implement the SIMPLE method, additional processing steps with dedicated applications implementing SIMPLE are required, thereby adding to the complexity and fragility of a system.

4.4 Summary and Discussion

In this chapter, we have defined the requirements for the integration of lineage with a digital object. We have presented a method that allows the integration of arbitrary metadata with digital objects and defined its realization to the byte level; the metadata we propose to embed is integrity and lineage metadata. As a foundation for the establishment of lineage, we use cryptographic hash functions on the digital objects for their unambiguous identification. Additionally, we have presented a technique that allows changes to be detected in the digital object's data or metadata relying only on contents of the digital object itself.

In the remainder of the chapter, we discuss the decisions taken in the SIMPLE method and evaluate SIMPLE against the requirements as defined in the first part of this chapter (section 4.1). We will conclude with some general issues. The requirements are as follows.

- Identify Digital Objects Independent of Information Systems
- Inhibit Digital Object – Metadata Dissociation
- Simplify Metadata Identification and Access
- Secure Integrity
- Make Lineage Relationships Explicit
- Accommodate Arbitrary Metadata
- Invariant Towards Software and Technology
- Minimize Complexity

The integration of XML metadata and digital objects satisfies the defined requirements "Inhibit Digital Object – Metadata Dissociation", "Simplify Metadata Identification and Access", "Accommodate Arbitrary Metadata", and "Minimize Complexity." Since the metadata of the digital object is integrated in the digital object, the metadata will remain in whatever information system in which the digital object is managed. The metadata can only be dissociated from the data if the digital object is changed. Metadata discovery is made possible by a defined byte sequence that also identifies the metadata as metadata, thereby aiding any human with a knowledge of English in classifying the encoded information. The well-defined encoding of the metadata makes it possible to discover and read the metadata of a digital object, even if the digital object format is no longer intelligible. The access and navigation within the metadata depends on the expected or defined metadata in the digital object, since we allow all metadata that can be encoded in XML as content of the metadata container. To minimize complexity we rely firstly on XML, *the* standard for vendor-independent information exchange; we profit from the verbosity of XML that makes it almost self-describing, provided that this requirement is also taken into account by the designers

of the additional metadata held by the metadata container. However, the dependence on XML counteracts the defined requirement "Invariant Towards Software and Technology" since there is clearly a dependence on XML technology. Nevertheless, to date there is no other technology which as wide spread, vendor- and platform-independent, and flexible. Although XML will be superseded by another technology some day, its intrinsic simplicity (if used correctly) and the availability of converters which exist even to date will make it possible to transform XML data into any other form of data needed. XML has a built-in exit strategy: its readability and intelligibility for humans.

The requirements "Identify Digital Objects Independent of Information Systems" and "Secure Integrity" are both addressed by using cryptographic hash functions on the digital object. The identity of a digital object and its integrity are closely related in our approach, since with our definition of equality, changes in the digital object are reflected in its identity. This amalgamation of identity and integrity enables every user to assess the integrity and identity of any digital object from its identifier; the test for the integrity is reduced to the computing of the hash function and comparing it to the sought identifier. Conversely, an identifier of a digital object can easily be computed. The support for multiple hash functions increases the resistance against possible cryptographic attacks. Additionally, the sealing process add cryptographic hash values to the SIMPLE metadata that allow the integrity of the digital object to be assessed without the prior knowledge of its identifier. This enables integrity checks on the digital object level. However, this does not protect the digital object against malicious breaches of integrity, confirming only that the changed digital object retains its integrity but not its identity. A stealthy malicious breach will re-embed the specific cryptographic hash values for the changed digital object called for by the SIMPLE method. However, since the digital object identity is determined by cryptographic hash functions the identity of the changed digital object will not be the same as that of the unchanged one.

The requirement "Make Lineage Relationships Explicit" is implemented by adding lineage metadata to the SIMPLE XML metadata container. The lineage metadata references the digital objects used in the creation of the digital object (its ancestors) recursively, that is, the ancestors reference their ancestors, which again reference their ancestors, and so on, to the creation of the primal digital objects, which have no ancestors. The lineage as defined only accounts for digital objects. However, processes creating digital objects often rely on additional inputs other than digital objects, like a human input, either in the form of a decision in a workflow or creative input in writing a document. In our model, input other than digital objects is ignored, since we do not know of any system, model, or framework that allows the description of such input, and the creation of such a system is clearly beyond the scope of this thesis. Another issue we do not define is the extent of the lineage which is embedded in the digital object or its level of detail. However, since the lineage uses identifiers as defined above, the lineage can be recursively reconstructed if the referenced digital objects are at hand. The addition of identifiers of ancestors to the lineage of a digital object aids in securing the integrity of the ancestors, since the identifiers

are distributed with the digital object, thereby making it difficult for an attacker to change ancestors without changes being noticed. Even if an attacker succeeds in manipulating a digital object along with its identifier in a certain environment without being noticed, all digital objects derived from the previously unchanged digital object will still bear the original identifier from the now manipulated digital object thereby increasing the probability of detecting the manipulation.

There may be privacy concerns when every manipulation of a digital object is recorded and all intermediate digital object byproducts of the process are referenced. This is even more so if the digital object (or better: its descendant) is transformed in various processes in different information systems and its transformations in these systems can be traced. These fears can be rebutted by pointing out that the control of which information is stored in the digital object lies with the systems manipulating the digital object, and therefore, ultimately, in the hand of the user. Additionally, the hash values used as identifiers cannot be used to determine the contents of the digital object due to the irreversibility of the hash algorithms.

We will end this discussion with a more general view of the SIMPLE method and its motivation. The digital object and its ancestors are the focus of this approach, since we see that to date there is no easy way to reference digital objects or any digital data independent of an information system. More importantly, there is no way to reference and thereby attribute digital objects to digital objects. Even though much of today's output from science, administration, industry, and leisure is stored in digital form, it is still not possible to reference a specific entity or a digital object independently of the information system managing it. Additionally, the copying of digital objects is inherently simple and is even endorsed, be it only for backup reasons, adding to the dissemination of the digital objects. The SIMPLE method provides information system independent identification, and attribution for digital objects and their metadata, with verifiable integrity.

In the next chapter, we present a prototypical implementation of the SIMPLE method to support our claims.

5

SIMPLE Prototype

In this chapter, we present a prototype that satisfies the requirements set out in the previous chapters, implementing the SIMPLE method discussed in section 4.2. Even though the SIMPLE prototype is the subject of this chapter we must stress that all functionality covered by the prototype can be implemented with standard tools that support appending of files like the unix `cat` or the windows `copy` command, text editors, and hash value generators. However, implementing SIMPLE with these basic tools is very error-prone and unduly burdensome for any but the simplest tasks.

5.1 Functionality

The functionality of the SIMPLE prototype covers all areas of the SIMPLE method. We have decided to break up the functionalities needed for the prototype into the following blocks: *metadata generation*, *metadata integration*, *digital object sealing*, *metadata identification and extraction*, and *integrity verification*. Figure 5.1 shows the modules and their inputs and outputs. In the following subsections the functionalities will be broken down and related to the requirements of the SIMPLE method. The decomposition follows the steps needed to manually create a SIMPLE digital object where every module builds on the results of the previous module.

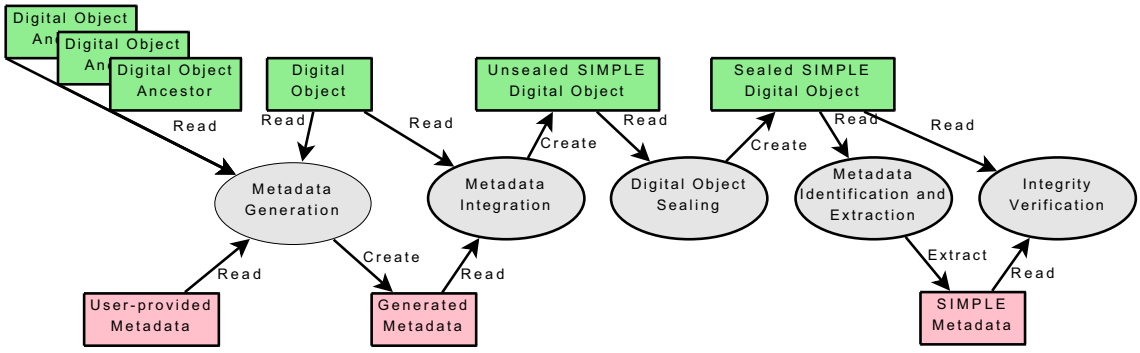


Figure 5.1: Dependencies between the modules of the SIMPLE prototype and the module's in- and outputs

To illustrate the features of each module, we will underline the functionality of each module with an example. We have chosen the creation of a SIMPLE digital object from a digital object in the setting of the State Archives of Appenzell Ausserrhoden as an overarching example. The original digital object is a resolution from the cantonal executive government ("Regierungsratsbeschluss"), a file in PDF format.

5.1.1 Metadata Generation

In this subsection, we describe the metadata generation functionality of the SIMPLE prototype. The functionality coveres all metadata to be embedded with the digital object. The metadata generated or included in this functionality falls into three categories: The arbitrary metadata to be included with the digital object; the identification and ancestry metadata identifying the digital object and its ancestors; and the integrity metadata, which is prepared for the sealing process. The arbitrary metadata includes user-provided metadata as well as process metadata from the SIMPLE prototype. The process metadata is included to ensure the traceability of the generation and embedding of the metadata with a digital object. Since the output of this functional block is XML metadata, any user-provided metadata included must be valid XML. This validity is checked automatically. The traceability of processes is further discussed in [Heuscher, 2006].

All inputs are digital objects, be it the digital object for which the metadata is destined, or its ancestors. Metadata identifying these digital objects is added to the ancestry metadata of each digital object. This includes path, file name, and access time for digital objects stored as files. These metadata attributes were chosen because they are important for identifying a file within the file system. This traceability metadata serves as a starting point for retracing the digital object's history within the file system. Additional file system attributes like file creation time, user and group rights, or last access time could be included. For the sake of brevity, we decided not to include more attributes in the implementation. Only files in a file system are supported as digital objects by the SIMPLE prototype.

Additionally, the automatically generated metadata is enhanced with a natural language text explaining the rationale and the contents of each XML structure. The language of these texts can be chosen as English or German.

Integrity metadata is the metadata that will be completed with the sealing of the digital object. This metadata is prepared with the "dummy" hash values, which will be replaced with values that allow the assertion of integrity. For better traceability, the original values of the "dummy" hash values are also recorded with the integrity metadata. This, together with the embedded documentation on the sealing process, will allow integrity checks, even if the details of the SIMPLE method's sealing process are no longer known.

For the accompanying example, this is the most complex step, since all inputs must be defined and all initial metadata is created in this step. The inputs for this step are a PDF file

```
<staatsarchivAR xmlns:dc='http://purl.org/dc/elements/1.1/'>
   <dc:source>Staatsarchiv Appenzell Ausserrhoden</dc:source>
   <dc:identifier>[replace with original filename]</dc:identifier>
</staatsarchivAR>
```

Figure 5.2: User-provided XML metadata template to be included into the SIMPLE XML metadata

`c:\temp\RRB-2000-021.pdf` and the user-provided metadata (figure 5.2). This metadata defines a Dublin Core XML namespace and a placeholder `[replace with original filename]`, which is automatically replaced with the filename of the original digital object.

Now the metadata is created from these inputs. We will use XML code snipplets to explain our implementation of the SIMPLE XML metadata. The XML metadata snipplets have been wrapped and indented to make them printable and easier to read.

The first element is the `metadata` element, which also serves as "magic number" for identifying the SIMPLE metadata within a digital object. As this element is part of the SIMPLE structure, it contains a `documentation` attribute which provides a natural language description (we chose English) of the contents and meaning of the element. We therefore let the first element speak for itself (figure 5.3).

```
<metadata describes="" documentation="This XML document, contains
      additional information (metadata) on this digital object (this
      file). The definition of the metadata follows the SIMPLE concept,
      which allows the integration of arbitrary data that describe the
      digital object. The attribute *describes* defines what is being
      described, the empty content signifies that this digital object
      including this SIMPLE metadata are being described.">
```

Figure 5.3: First part of the SIMPLE XML metadata

Next, the user-provided metadata is embedded in the SIMPLE metadata and the placeholder is replaced with the associated content. Here, the placeholder "`[replace with original filename]`" is replaced with the filename of the original digital object "`RRB-2000-021.pdf`." No additional natural language description is provided, since there was no additional description in the user-provided metadata (figure 5.4). The XML entities `
` stem from line breaks

```
<staatsarchivAR xmlns:dc="http://purl.org/dc/elements/1.1/">&#10;
  <dc:source>Staatsarchiv Appenzell Ausserrhoden</dc:source>&#10;
  <dc:identifier>RRB-2000-021.pdf</dc:identifier>&#10;
</staatsarchivAR>
```

Figure 5.4: User-provided part of the SIMPLE XML metadata with expanded placeholder

of the user-provided metadata, which have been replaced with their corresponding document entities to make the resulting XML span only one line. This enables the integration in line-based interpretation environments like shell scripts.

The `identification` element is again part of the automatically generated SIMPLE metadata. It holds all information relevant for the identification of the SIMPLE digital object. In our implementation, this includes information on the file for which the SIMPLE digital object will be written as well as information on the identification of its ancestors (figure 5.5). Note that the `ancestor` element again uses the `identification` element for the identification of an ancestor. In this way the identification part of the SIMPLE XML can be re-used when creating a SIMPLE digital object from one or more digital objects which contain SIMPLE metadata. Also note that only the ancestors provide a hash value for the identification, since hash values of the SIMPLE digital object are not yet known and would become obsolete if they were to be written into the newly created digital object.

```
<identification documentation="This element details how the digital
      object can be identified.">
  <file currentDateTime="2008-12-15T17:36:29.506 CET"
      >C:\temp\RRB\RRB-2000-021.simple.pdf</file>
  <ancestors documentation="This element details from which digital
        objects the digital object was derived.">
    <identification documentation="This element details how the
          digital object can be identified.">
      <md5 documentation="MD5 algorithm according to RFC 1321."
          >8efc764020eac57b5e451114cfd6f023</md5>
      <sha1 documentation="SHA1 algorithm according to RFC 3174."
          >4d571ed5b5471e463277b911c31b570f14899a4e</sha1>
      <file currentDateTime="2008-12-15T17:36:29.521 CET"
          >C:\temp\RRB\RRB-2000-021.pdf</file>
    </identification>
  </ancestors>
</identification>
```

Figure 5.5: Identification part of the SIMPLE XML metadata

The `integrity` element holds its own documentation, in which the process of verification is described. The documentation is not yet correct, since the sealing process has not yet taken place. However, the corresponding elements for the sealing process have already been added (figure 5.6). They are identified by the attribute `covers` with the value `data metadata`. The elements containing the attributes have the same textual content as the content of their `dummyValue` attribute, which is a prerequisite for the sealing of the digital object.

```
<integrity documentation="The integrity of this digital object was
      secured with mechanisms that will be described later. To verify
      the integrity of this digital object, replace the text within
      the following elements with the content of the *dummyValue*
      attribute. Then calculate the hash value of the changed digital
      object and compare it with the replaced value of the element. If
      the replaced value and the calculated values are equal, the
      digital object has not been altered. The attribute *covers*
      defines which parts of the digital object were used to calculate
      the hash values (where *data* means that only the data part was
      used and *data metadata* means that all of the digital object
      was used).">
  <md5 covers="data"
      documentation="MD5 algorithm according to RFC 1321."
    >8efc764020eac57b5e451114cfd6f023</md5>
  <sha1 covers="data"
      documentation="SHA1 algorithm according to RFC 3174."
    >4d571ed5b5471e463277b911c31b570f14899a4e</sha1>
  <md5 covers="data metadata"
      documentation="MD5 algorithm according to RFC 1321."
      dummyValue="................................"
    >................................</md5>
  <sha1 covers="data metadata"
      documentation="SHA1 algorithm according to RFC 3174."
      dummyValue="........................................"
    >........................................</sha1>
</integrity>
```

Figure 5.6: Integrity part of the SIMPLE XML metadata

The `creation` element holds process metadata for the creation of the SIMPLE digital object. Note that the `software` attribute contains versioning information on the software used. Here, we provide information about the software version used in the creation of the SIMPLE digital object as well as specific versioning information about the core components. Additionally, information on how to reverse the process is embedded in natural language in the `undo` attribute. For a reversing process to be automated, the contents of this attribute will need to be formalized. This is, however, left for future work. The `environment` element contains a minimum set of information about the environment in which the SIMPLE digital object was created (figure 5.7).

We have tried to keep the information as concise as possible. Despite our efforts, the full SIMPLE XML is still of considerable size (about 4000 bytes).

```
  <creation date="2008-12-15T17:36:29.584 CET"
      documentation="This element details how this digital object was
        created. The attribute *date* details when this digital object
        was created, according to the gregorian calendar. The attribute
        *software* details the software used in the creation of the
        digital object."
      software="SIMPLE Utilities (Java) Version 0.6.1; $Revision: 1.49 $,
        $Date: 2008/05/30 18:57:03 $, $Author: heuscher $">
    <modification documentation="This element details how the SIMPLE
          metadata was integrated into the digital object. The attribute
          *method* describes the method how the SIMPLE metadata was
          integrated into the digital object in clear text. The
          attribute *software* defines the software which integrated the
          SIMPLE metadata into the digital object. The attribute *undo*
          describes the method how the integration of the SIMPLE
          metadata can be undone in clear text. "
        method="append the xml metadata at the end of the data"
        software="ch.heuscher.simple.embed.PdfMetadataEmbedder;
          $RCSfile: PdfMetadataEmbedder.java,v $, $Revision: 1.4 $,
          $Date: 2007/04/01 16:58:55 $, $Author: heuscher $"
        undo="remove this line including the new-line bytes"/>
    <environment documentation="This element details the environment in
          which this digital object was created.">
      <system property="java.vm.version">1.6.0_03-b05</system>
      <system property="os.name">Windows XP</system>
    </environment>
  </creation>
</metadata>
```

Figure 5.7: Process part of the SIMPLE XML metadata

5.1.2 Metadata Integration

In this subsection, we describe the metadata integration functionality of the SIMPLE prototype. The metadata generated by the metadata generation block is integrated with the digital object using this functionality.

Depending on the format of the digital object the metadata is integrated with the digital object. As proposed in subsection 4.2.3 the metadata will be embedded at the end of the digital object as comment where possible or needed. The format of the digital object is established using its suffix; an appropriate form of embedding is used for each supported format. All confined formats are supported by default, since the XML metadata can be appended without additional wrapping. The confined formats include the majority of the binary formats like JPEG, DOC, ZIP, and TIFF. The following open-ended formats are supported by the SIMPLE prototype: XML, HTML, Java, JavaScript, and PDF. The unsupported, open-ended formats are treated like confined formats, in that the metadata is appended without additional wrapping.

Additionally, a natural language text (specific to the digital object format) which explains how the embedding of the metadata can be undone, is added to the metadata, as well as traceability metadata as described in subsection 5.1.1. This metadata includes the time and date of the embedding of the metadata, the software version of the SIMPLE prototype and its integration component, as well as metadata on the computing environment used to integrate the metadata.

An additional functionality of this block is the reversing of the metadata integration, whereby the metadata is removed from the digital object. The resulting digital object is equal to the digital object prior to the metadata integration.

For the accompanying example, we will show how the metadata is added to the digital object to create a SIMPLE digital object. Since the prototype uses the *binary append* method to add the SIMPLE XML to the digital object, we will concentrate on the end of the PDF file to be converted (figure 5.8). For easier reading, the end of the PDF file has been converted to ASCII.

```
0000007086 00000 n
trailer
<</Size 6>>
startxref
116
%%EOF
```

Figure 5.8: End of the example PDF file before the integration of SIMPLE XML metadata

After integration, the end of the PDF file also marks the start of the SIMPLE XML. It is appended as PDF comment, which is defined as percent sign '%' at the beginning of a new line. The line break was also added by the metadata integration module in order to better separate the SIMPLE XML from the rest of the PDF file. The transition from the original PDF file to the SIMPLE XML (converted to ASCII) is shown in figure 5.9. The three fullstops . . . at the end are the placeholder for the rest of the SIMPLE XML, as shown in subsection 5.1.1.

```
0000007086 00000 n
trailer
<</Size 6>>
startxref
116
%%EOF
%<metadata describes="" documentation="This XML document, contains
additional information (metadata) on this digital object (this file).
...
```

Figure 5.9: Transition from the end of the example PDF file to the SIMPLE XML after the integration of SIMPLE XML metadata

5.1.3 Digital Object Sealing

In this subsection, we describe the digital object sealing functionality of the SIMPLE prototype. The metadata generated and integrated by the metadata generation block is enhanced to allow the later verification of the integrity of the digital object. This functionality calculates the identifiers of the unsealed SIMPLE digital object and inserts them at the predefined locations as defined in subsection 4.2.3.

For the accompanying example, we will show how the SIMPLE XML metadata is updated to conform to the sealing process. As a prerequisite, a digital object with the embedded SIMPLE XML metadata must be available. More specifically, the XML structure must contain an `integrity` element, that has at least one sub-element with the `covers` attribute of the value `data metadata`, and a `dummyValue` attribute which has the same value as the content of the sub-element. The name of the sub-element defines the hash algorithm to be used. Figure 5.10 shows the contents of the elements prior to the sealing process.

```
<md5 covers="data metadata"
    documentation="MD5 algorithm according to RFC 1321."
    dummyValue="................................"
               >................................</md5>
<sha1 covers="data metadata"
    documentation="SHA1 algorithm according to RFC 3174."
    dummyValue="........................................"
               >........................................</sha1>
```

Figure 5.10: Extract of the integrity part of the SIMPLE XML before the sealing of the SIMPLE digital object

The hash value of the digital object with the embedded SIMPLE XML is determined for each of the hash algorithms defined by the SIMPLE XML. This hash value is then embedded in the corresponding element. Figure 5.11 shows the contents of the elements after the sealing process.

```
<md5 covers="data metadata"
    documentation="MD5 algorithm according to RFC 1321."
    dummyValue="................................"
               >93684be08d329429a4f41a1156e769d4</md5>
<sha1 covers="data metadata"
    documentation="SHA1 algorithm according to RFC 3174."
    dummyValue="........................................"
               >8e1e98b9af37d720aeef7fe87912485e8b631c4b</sha1>
```

Figure 5.11: Extract of the integrity part of the SIMPLE XML after the sealing of the SIMPLE digital object

After the sealing process, the creation of the SIMPLE digital object is completed. Its integrity can now be assessed independently of a specific information system.

5.1.4 Metadata Identification and Extraction

In this subsection, we describe the metadata identification and extraction functionality of the SIMPLE prototype. This functionality allows the discovery of all SIMPLE metadata within digital objects and the extraction of this metadata.

SIMPLE metadata within a digital object is identified and extracted to be used further independent of the format of the digital object. Possible uses include the rendering of the metadata for human interpretation, the propagation of ancestry metadata to other digital objects, or the verification of the integrity of the digital object. The digital object is not changed in the process.

The SIMPLE metadata is identified by scanning the digital object for the sequence described in subsection 4.2.2 on the byte level. When the sequence is found, the end sequence is sought, and if found the metadata is extracted. Multiple instantiations of SIMPLE metadata can be found within one digital object in this way; all instances are extracted and can be used for further processing.

For the accompanying example, the resulting SIMPLE XML is identical to that of the example in subsection 5.1.1, except that the contents of the elements in figure 5.6 have been replaced with those of figure 5.11.

5.1.5 Integrity Verification

In this subsection, we describe the integrity verification functionality of the SIMPLE prototype. This functionality allows the testing of the integrity of a digital object which has been sealed. This functionality undoes the sealing process, calculates the identifiers of the unsealed SIMPLE digital object, and compares the calculated identifiers with those provided by the SIMPLE metadata. If the identifiers match, the integrity is intact, otherwise the integrity is compromised.

For the accompanying example, the result of the verification process will be positive, under the premise that the SIMPLE digital object has not been changed.

5.2 Implementation

In this section, we highlight the decisions taken in the implementation of the SIMPLE prototype.

5.2.1 Technologies

We chose Java as our implementation technology because of its platform independence and pervasiveness. The Java virtual machine version 1.4 [Sun Microsystems Incorporated, 2008] was chosen as a target platform, since this was the first version to include libraries for XML processing

and, although introduced in 2002, this version is still widespread. Additionally, the hash function implementations *MD5* and *SHA-1* are included as libraries. All functions needed are exported as part of a library which handles the processing of the digital object and its metadata. For the SIMPLE prototype, digital objects are always files in a file system. The web interface utilizes JavaServer Pages (JSP) technology version 1.0 [Pelegrí-Llopart et al., 1999]. This allows the web interface to be run on the first application servers supporting JSP and all application servers thereafter. The web page technology used is XHTML 1.1 [McCarron and Ishikawa, 2007], CSS 2.1 [Bos et al., 2007], and ECMAScript for XML (E4X) [Schneider et al., 2005].

5.2.2 Interfaces

As the core of the functionality of the SIMPLE prototype lies in the library, different interfaces can be implemented with relative ease. Three interfaces were implemented to support this claim. First, we have opted for the implementation of a web interface with limited use of the SIMPLE core library, implementing only one workflow to integrate metadata with a digital object. However, the full functionality of creating a SIMPLE digital object from a file is implemented in the web prototype. This interface helps in demonstrating the SIMPLE method without the need for downloading or installing additional software.

The second interface to the library is a command line wrapper which exposes the full functionality of the library at command line level. This allows the integration of SIMPLE in a workflow type environment and operating system automation like shell scripts.

The third interface is an automatic build tool interface ([Apache Ant, 2008]), which allows the automated processing of a large number of digital objects and provides only the limited functionality of adding metadata and sealing the digital object.

The web interface implements the functionality of checking the integrity of digital objects, adding metadata to digital objects (which are uploaded and shared), and the possibility to download digital objects, with or without their integrated metadata, as well as downloading only the metadata. On upload, the user can choose the language of the documentation added to XML entities and the identification functions to be used with the digital object (*MD5* and/or *SHA-1*). The user also provides metadata in XML format, which he can enter as text, as well as the digital object (file) he wishes the metadata to be added to (figure 5.12).

The command line interface provides the full functionality of the SIMPLE library to the user. It is implemented as a wrapper around the SIMPLE prototype library, with a utility function to calculate hash values of a digital object. The implemented functions are as follows:

- Add automatic metadata and seal: This function covers all of the SIMPLE workflow shown in figure 5.1 except for the checking of the integrity. The inputs of this function are the digital object, the user-provided XML metadata, the identification functions (which are also

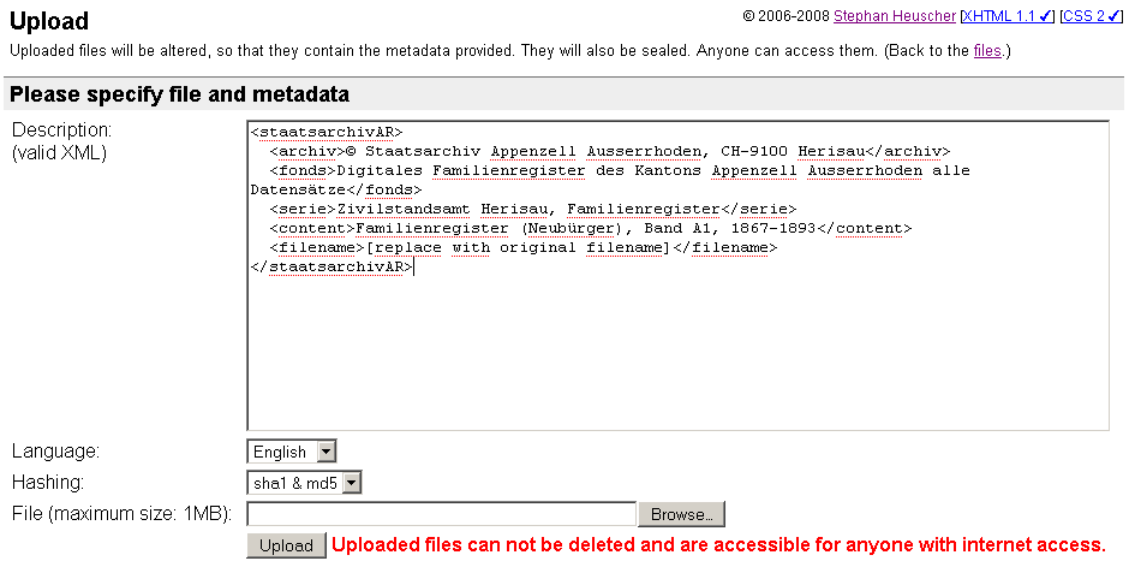


Figure 5.12: Screenshot of the upload interface of the SIMPLE prototype

used for sealing the newly created digital object), and the ancestors of the digital object. As output all SIMPLE metadata (comprised of user-provided, lineage, and integrity metadata) is automatically created, and that metadata is integrated with the digital object, thereby creating a new digital object. This new digital object is sealed so that changes can be detected thereafter. This function is useful for executing all steps in the SIMPLE workflow at once.

- Insert metadata: This function integrates the provided metadata with the provided digital object. No additional metadata is created, nor are any tests applied to the integrated metadata. The creator of the metadata needs to ensure compliance with the SIMPLE requirements. The function is useful if the SIMPLE XML or other metadata is created independently of the SIMPLE prototype and needs to be integrated with the digital object.

- Extract metadata: This function extracts the SIMPLE XML metadata and stores it independently of the provided digital object. The function is useful when the SIMPLE metadata of a digital object needs to be processed or viewed independently from the rest of the digital object.

- Calculate hash values: This utility function calculates the hash values of a provided digital object. The hash algorithms or identification functions are defined by the user. The function is useful because the user will not always have other tools to create or verify identifiers of digital objects.

- Check integrity: This function checks the integrity of the provided SIMPLE digital object. The function is useful because it automates the integrity check of a SIMPLE digital object.

- Seal digital object: This function seals the digital object as described in subsection 4.2.3. The function is useful because it allows digital objects to be sealed, irrespective of how the unsealed SIMPLE metadata was integrated with the digital object.

The automatic build tool interface of the SIMPLE prototype is implemented for the Apache Ant build environment, and defines one Ant task which provides the same functionality as the upload function of the web interface, but for a bulk of digital objects. The task integrates user defined metadata with the defined digital objects, for example all files in a directory tree.

5.2.3 XML Implementation

The basic structure of the metadata is provided by the definition of XML as storage technology, by the definition of the "metadata" element as an XML root element, and by its encoding as defined by the SIMPLE method. However, the other parts of the SIMPLE metadata are not defined. Therefore mappings and implementations for the ancestry, integrity, and arbitrary external metadata need to be defined, with respect to the traceability and documentation requirements. In this subsection, we will present the implementation decisions made to represent the SIMPLE metadata as XML.

Structure and Validation No DTD or XML schema exists that defines the structure of SIMPLE XML. This is a deliberate decision since the structure of the metadata is only exemplified in the SIMPLE prototype; it serves as a starting point, not as a final version. The SIMPLE XML document must be well-formed according to the XML specification [Bray et al., 2004].

XML metadata included in the SIMPLE XML may define its own schema or namespace. However, the included metadata is not tested against its schema.

Documentation In order for the stored SIMPLE XML metadata to be intelligible for a long period of time, additional descriptive natural language information about the rationale of the metadata presented is embedded with the metadata. Therefore, every element of the SIMPLE XML metadata is enhanced by a short natural language description of the purpose of the element. This documentation is embedded in the SIMPLE XML as value of the attribute *documentation*. For an example of the English documentation see figure 5.13. The language of this documentation can be chosen by the user when creating the metadata. The SIMPLE prototype supports the languages English and German.

Arbitrary Metadata SIMPLE XML embeds any valid XML metadata. This metadata is embedded just after the root element since it may be more important for the user than the rest of the automatically generated metadata. All other SIMPLE metadata is appended after the arbitrary metadata.

```
<creation documentation="This element details how this digital object
was created. The attribute *date* details when this digital object was
created according to the Gregorian calendar. The attribute ...">
...
</creation>
```

Figure 5.13: English natural language description of the "creation" SIMPLE XML element

Identity and Ancestry Metadata Every digital object can be identified using an identification function. We use cryptographic hash functions in the SIMPLE prototype. However, the result of the identification function cannot be embedded with the digital object or the SIMPLE metadata (being part thereof) without changing the result of the identification functions. Therefore, the identification is inherent to, but not integrated with, the digital object. With the SIMPLE prototype, any metadata supporting the identification of the digital object can be embedded with the identification metadata, like a digital object identifier, a file name, or a textual description of the digital object. Since the SIMPLE prototype supports file system information systems, we add the file name and the path of the created digital object to its identification metadata. This does not preclude that the SIMPLE digital object must be accessible at this path, only that the digital object was accessible at this path at a certain point in time.

Since the identification of a digital object cannot be embedded, additional metadata to define the identification of a digital object must be used. We chose the lineage metadata as refining metadata for the identification because all ancestors of the digital object contribute to the identity of the final digital object. The lineage of a digital object can therefore be seen as part of its identity, an approach which we pursue in the SIMPLE prototype. The ancestors of the digital object, which form part of the identification of the digital object, are again identified using the same metadata structure. The identification metadata for the direct ancestors is enhanced with the results of the identification function on the ancestor, for example the value of applying the SHA1 identification function on the ancestor.

The identification metadata includes all currently known identification metadata from the ancestors of the digital object, which recursively includes all identification from its ancestors. The ancestry metadata of a digital object in the SIMPLE prototype is restricted to lineage metadata, that is, only metadata of digital objects contributing to the creation of the digital object is used. Metadata on other digital objects created from the ancestors is not used. This representation of ancestry metadata is well suited for mapping to XML, since the inherent structure is similar to the hierarchical structure of XML. The hierarchical structure is defined by a root (the digital object or the XML root element) with hierarchical descendants (digital object ancestors or XML child elements).

5.3 Evaluation, Advantages and Limitations

In this section, we compare the specifications of the SIMPLE method to its implementation in the prototype and discuss the advantages and limitations of the prototype. The main advantage of the SIMPLE prototype is its automation of laborious manual tasks to create and integrate metadata with digital objects, to identify and extract SIMPLE metadata, to seal digital objects, and to verify the integrity of sealed digital objects. The automation enables the integration of the SIMPLE method into workflows. As this is a prototype, limitations are clearly perceived. They range from a small set of supported open-ended digital object formats over an incomplete web interface to the definition of the structure of the SIMPLE XML metadata.

This section is divided into three parts. First, we look at the SIMPLE prototype on the digital object level; second, at the metadata level; and third, we look at possible enhancements of the SIMPLE prototype.

5.3.1 Digital Object

In this subsection, we focus on the implementation of the method on the byte level, which is the digital object level. On this level, the SIMPLE method calls for

1. Identifiable metadata: The beginning of the metadata is clearly defined at byte level, so it can be detected easily. This requirement has been implemented.

2. XML encoded metadata: All metadata has to be encoded as valid XML. This requirement has been implemented.

3. Integration of metadata: The metadata should be integrated as comment for open-ended digital object formats or appended at the end of the digital object for confined digital object formats. This requirement has been implemented. However, the detection of open-ended digital object formats and their corresponding integration of the metadata is limited to a number of commonly used formats.

4. Sealing: The metadata is prepared for the inclusion of the hash values that secure the integrity of the digital object, after which the actual values are computed on the bytes of the newly created digital object and, finally, inserted. This requirement has been implemented.

On the byte level there are no additional requirements. Other requirements imposed by the SIMPLE method concern the metadata, which we will discuss next.

5.3.2 Metadata

In this subsection, we focus on implementation at the metadata level. At this level the SIMPLE method calls for

1. XML structure: The SIMPLE method defines the root element (metadata) and one of its attributes (describes). This requirement has been implemented.

2. Arbitrary metadata integration: Any metadata must be integrated in the metadata. This requirement has been implemented for metadata in XML form. However, the SIMPLE prototype will not distinguish the integrated metadata from the automatically generated metadata and will therefore not be able to discern an end tag from the SIMPLE metadata from the same end tag in the integrated metadata: a *</metadata>* tag in the integrated metadata will be falsely identified as an end tag of the SIMPLE metadata, for example.

3. Explicit ancestry relations: The ancestry relations are implemented as part of the identification metadata, recursively identifying ancestors. This requirement has been implemented.

4. Natural language documentation: It is possible to add a natural language description explaining the rationale behind the element, its child elements, and its attributes for each element in the metadata. This requirement has been implemented.

Since the structure of the SIMPLE XML metadata is not defined, we chose to keep the structure basic and self-explanatory. The structure is however only defined by the SIMPLE prototype.

5.3.3 Possible Enhancements of the Prototype

Being a prototype the application has many areas in which it still needs to evolve.

The stability of the implementation has not been determined. For an industrial strength application, the automatic test would need to be extended, especially in the light of the possible differences between the different implementations of the Java virtual machine and the respective XML libraries.

Digital objects are currently always local files. This constraint should be dropped to allow all kinds of digital objects to be processed by the SIMPLE prototype. This would include remote resources as well as digital objects in database management systems.

The structure of the SIMPLE metadata is currently undefined. For the definition of this structure, the proposition made by the SIMPLE prototype must be critically discussed by users of SIMPLE with different backgrounds.

The non-command line interfaces could be adapted for the user interface to provide the full functionality of the SIMPLE prototype library. This may also include a navigation through the

ancestry data for identifying and accessing ancestors of the digital object. This can however only be implemented as part of an information system spanning the possible ancestors.

From a software engineering point of view, the library of the prototype should be constructed in a more modular way, possibly using a plug-in mechanism for third-party extensions. More specifically, the functional blocks described in section 5.1 would benefit from clear interfaces and defined interactions between them.

Currently, only a small number of open-ended digital object formats are supported by the prototype, and the identification of the format is based solely on the file name suffix. Using existing format identification tools like [JHOVE, 2008], the identification of formats would no longer be constrained to files. Additionally, the inclusion of third-party metadata integrators would augment the flexibility of metadata integration.

We see the definition of the granularity in which the ancestry metadata is used as another possible enhancement. The current implementation only aggregates the lineage metadata from the direct ancestors, without any possibility for the user to restrict or refine the ancestry metadata.

Reaching further, the performance of the SIMPLE prototype could certainly be improved. However, the tested uses of the prototype did not show the need for such an improvement.

5.4 Case Study

To critically review the claims set forward by this thesis for the SIMPLE prototype, the State Archives of Appenzell Ausserrhoden have agreed to include the SIMPLE method into the upcoming migration of their scanned family records. In this section, we will present the migration and the use of the SIMPLE method and prototype.

5.4.1 State Archives of Appenzell Ausserrhoden

The State Archives of Appenzell Ausserrhoden, in collaboration with the library of Appenzell Ausserrhoden, ensure the persistent documentation of the canton of Appenzell Ausserrhoden. They store and index all records of lasting value for the administration and court of justice in the canton of Appenzell Ausserrhoden. The State Archives also accession records which are of high value to the history of the canton from public institutions or private holdings (associations, companies, families).[1]

The State Archives of Appenzell Ausserrhoden was chosen for the case study because it is very active in the area of digital long term preservation, for which their first experiences date back to the year 2000. They are considered as small archives; the current staff consists of three persons.

[1] This paragraph is a translation from [Appenzell Ausserrhoden, 2008].

5.4.2 Goal

The goal of the migration is to traceably migrate scanned pages of the family records of the canton of Appenzell Ausserrhoden from the PDF to the PDF/A [ISO, 2005] format. The PDF/A format was specifically designed for long term archive use. Additionally, metadata designed by the State Archives was to be added to each scan to aid in later genealogical requests prompted by the use of the family records. Today, many requests provide photocopies of pages taken from the family records. These photocopies do not provide sufficient metatata to allow the identification of the exact register, thus making research highly inefficient.

The goal for the SIMPLE prototype is to demonstrate its ease of use, the advantages of the SIMPLE method (addition arbitrary metadata, verifiable integrity, and ancestry recording) in the environment of a small archives. The rationale behind choosing a small archive is that big archives should already have some system for securely storing digital objects and their metadata. These systems are not in use with small archives since they are too costly and labor intensive to set up and operate. Here, SIMPLE enables small archives to implement a simple solution for their metadata and integrity needs for digital objects.

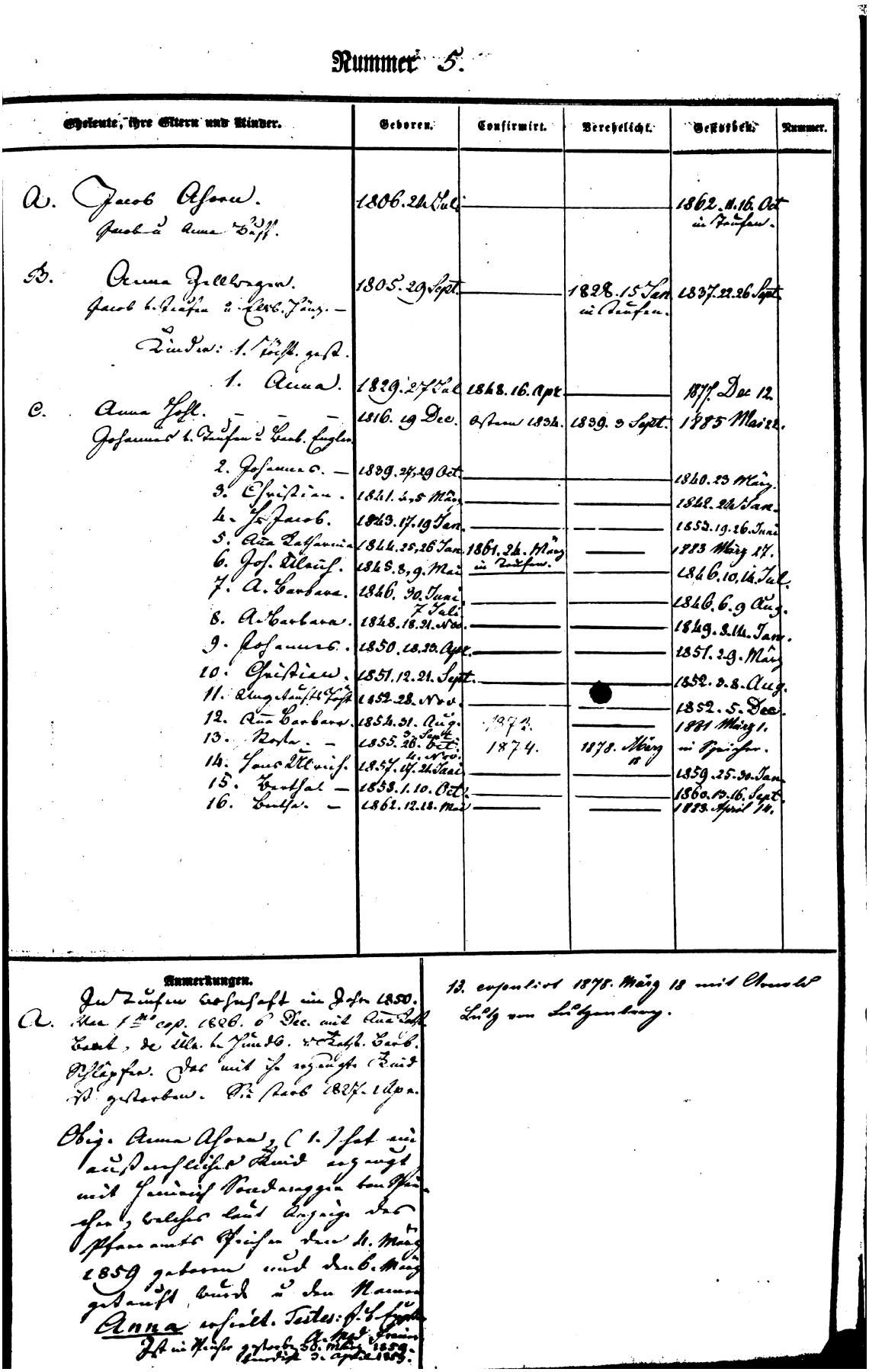
5.4.3 Setup

The scanned family records are stored on shared directories where each directory represents one volume of the family records as shown in figure 5.14. The conversion of the scans to PDF/A is conducted on a personal computer which runs Microsoft Windows XP as an operating system and a SUN Java runtime environment 1.4 as Java environment. The PDF is converted to PDF/A using *3-Heights^{TM}PDF to PDF/A Converter Shell* provided by PDF Tools AG. The SIMPLE prototype version 0.6.2 is installed on the personal computer for use with the command line interface. A short tutorial and hands-on introduction to the working of the command line interface of the SIMPLE prototype was provided for one employee of the archives.

The resulting output of the conversion (the converted PDF/A files including SIMPLE metadata) was to be stored on a shared directory, maintaining the file name of the original file. The main requirement was that the SIMPLE metadata must at least contain the file name of the original file.

5.4.4 Implementation

The process was automated using Windows scripting. Every directory, that is, every volume of the scanned family records, was processed separately. This automation level was chosen because the number of volumes is limited, and the additional metadata differed from volume to volume. The process involved the following steps:



Nummer 5.

Eheleute, ihre Eltern und Kinder.	Geboren.	Confirmirt.	Verehelicht.	Gestorben.	Nummer.
A. Jacob Asow. Jacob d Anna Daff.	1806. 24 Juli			1862. d. 16. Oct in Pusen.	
B. Anna Hellwagen. Jacob b. Pusen 2 Ehe. Jüng. — Kinder: 1. Todt geb.	1805. 29 Sept.		1828. 15 Jan in Pusen.	1837. 22. 26 Sept.	
1. Anna.	1829. 27 Juli	1843. 16. Apr		1877 Dec 12.	
C. Anna Jost. Johannes b. Pusen 2 Ehe. Jüng.	1816. 19 Dec.	d Anna 1834.	1839. 9 Sept.	1885 Mai 22.	
2. Johannes. —	1839. 27, 29 Oct.			1840. 23 Mai.	
3. Christian.	1841. 4, 5 Mai.			1842. 22. 1 Jan.	
4. Jacob.	1843. 17. 19 Jan.			1853. 19. 26. Juni	
5. Ana Catharina	1844. 25, 26 Jan.	1861. 24. März in Pusen.		1883 März 17.	
6. Jos. Klauf.	1845. 8, 9. Mai			1846. 10, 14. Jul.	
7. A. Barbara.	1846. 30. Juni 7 Juli			1846. 6. 9 Aug.	
8. A. Barbara.	1848. 18. 21. Nov.			1849. 8. 11. Jan.	
9. Johannes.	1850. 18. 20. Apr.			1851. 29. Mai	
10. Christian.	1851. 12. 21. Sept.			1852. 3. 8. Aug.	
11. Ungetauft todt	1852. 28. Nov.			1852. 5. Dec.	
12. Ana Barbara.	1854. 31. Aug.	1874.		1881 März 1.	
13. Rosa —	1855. 3. 26. Sept. Oct.	1874.	1878. März 18	in Preußen.	
14. Jane Klauf.	1857. 4. 24. Nov. Juni			1859. 25. 30. Jan.	
15. Rosalia —	1858. 1. 10. Oct.			1860. 15. 16. Sept.	
16. Lucia —	1862. 12. 18. Mai			1883. April 14.	

Anmerkungen.

A. Jacob Asow ist in Jahr 1850. den Pusen Gemeinschaft ausgetreten. Ana 1ste cop. 1826. 6 Dec. mit Ana Catharina Hellwagen, der Wwe. des Jacob Hellwagen. Das mit sie erzeugte Kind Anna ist gestorben. Die Mutter 1827. 1 Apr.

Obj. Anna Asow, C 1. Jost ist ein uneheliches Kind erzeugt mit Johann Vordenberg von Pusen, welches laut Anzeige des Pfarramts Pusen den 8. März 1859 geboren und den 6. März getauft wurde. Den Namen Anna erhielt. Testes: J. Jost, Anna Jost geb. Asow. Pusen, 6. März 1859. Kundgemacht d. 6. April 1859.

13. exportirt 1878. März 18 mit Ehemann Ludwig von Ludwigsburg.

Figure 5.14: Example of a scanned family record page (ZFR-14-B01-005)

1. Creation of metadata: For every volume, the metadata was created in the form of an XML file. See figure 5.15 for an example.

2. Conversion to PDF/A: All PDF files from the source directory were converted to PDF/A and stored in a temporary directory for further processing.

3. Adding of SIMPLE metadata: SIMPLE metadata was integrated with all PDF/A files. The metadata added also included the lineage of the PDF/A file, which stems from the original PDF file. For this, the original PDF file was accessible to the SIMPLE prototype. The result was stored in a second temporary directory.

   The command line interface of SIMPLE was used for this task since the build tool interface would have needed additional software to be installed on the personal computer which could have caused problems with the IT department. However, the performance of the SIMPLE prototype was still satisfactory, taking less than one second to integrate the metadata and seal each digital object.

4. Extraction of SIMPLE metadata: To allow straightforward access to the SIMPLE metadata, it was extracted from the newly created digital object and stored in a "metadata" sub-directory.

5. Testing of the result: The resulting digital objects were tested using random samples of digital objects from the process. Tests included the rendering of the converted files using a PDF viewer, the analysis of the SIMPLE metadata in the "metadata" sub-folder as well as checking that the metadata was integrated with the converted files using Microsoft Word. Microsoft Word was chosen since it can display the binary contents of digital objects and therefore also the SIMPLE XML metadata in each digital object. All tests were conducted manually by the archives' staff.

Additionally, all output of the automated procedure was captured in a log file. This file was checked for errors in the conversion to PDF/A and the subsequent steps.

5.4.5 Execution

The process described in subsection 5.4.4 was executed by a learned archivist who was given two hours of training and a trainee who was given instructions by the archivist. The migration was conducted in the period between December 2008 and September 2009 as a low priority background task. During this time the full migration was executed autonomously, without any support outside of the State Archives.

```
<staatsarchivAR>
  <archiv>
    (c) Staatsarchiv Appenzell Ausserrhoden, CH-9100 Herisau
  </archiv>
  <fonds>
    Digitales Familienregister des Kantons Appenzell Ausserrhoden
  </fonds>
  <serie>Zivilstandsamt Herisau, Familienregister</serie>
  <content>Familienregister, Band B1, 1829-1840</content>
  <filename>[replace with original filename]</filename>
</staatsarchivAR>
```

Figure 5.15: Example for user-defined metadata template for one volume of the scanned family records

5.4.6 Results

The migration was conducted as planned. To date, all 103'804 pages of the family records have been migrated. The PDF/A conversion tool detected some irregularities in the PDF source files, but all files could be converted without any visible changes. The adding and extraction of the SIMPLE XML worked flawlessly. The process described in subsection 5.4.4 for one directory of about 500 files with an average size of 100 Kb took about 6 minutes. The users commented on a few occasions that the SIMPLE method was easy and intuitive to understand.

5.4.7 Evaluation

The evaluation of the use of SIMPLE was conducted in the form of two interviews, one with the State archivist and one with the archivist responsible for the migration. The questions asked can be found in chapter A.

The case study shows that the SIMPLE prototype can be put to productive use in a small archives. The prototype was easily integrated into an operating system based automation environment (Windows scripting), and could be operated by an archivist with no IT background and a trainee after two hours of staff training. The IT department of Appenzell Ausserrhoden agreed that the SIMPLE prototype could be installed on any workstation without any further explicit consent and that the archivists could operate the prototype as they see fit. Staff members of the IT department were very happy that they were not burdened by having to maintain and support yet another application.

As a result of the migration, the migrated scanned pages integrate their lineage so that the original scan can always be identified, for example if the conversion to PDF/A would turn out to be erroneous. Additionally, and most importantly for the archives' staff, whenever a scanned page from their holdings is presented, they can identify the page without having to open the page

in a PDF viewer, since the information is in the file. Every file now has metadata associated to it that states its origin with the states archives.

State Archives staff that employed SIMPLE voiced drawbacks and suggestions as follows.

1. Usability of the Command Line Interface: The command line interface (DOS-Prompt) discourages the use of SIMPLE for non-IT users.

2. Hiding of Metadata in the Digital Object: When a digital object with SIMPLE metadata attached leaves the states archives, the user will not know about the additional metadata hidden in the digital object. This needs to be communicated since it is not self-explanatory.

3. Possible Deletion of Ancestor Digital Objects: For economical reasons, lineage may lead to false security that ancestor digital objects may be deleted, since the current digital object may be stated sufficient for economical reasons.

4. SIMPLE could be too Simple for Complex Digital Archives: Complex and bigger digital archives could fend off the SIMPLE concept because it appears too simple and therefore not usable in a complex scenario.

5. Integration with Archival Data Management: The archival data management could be integrated with the SIMPLE prototype so that the existing metadata would be taken from that system. To date, the metadata is provided manually per volume.

6. Situation of SIMPLE with Respect to the OAIS RM[2] Information Packages: It is unclear how to map a digital objects with SIMPLE metadata to an OAIS information package.

7. Inability to Track What Changed in a Digital Object: SIMPLE holds the lineage and can check the integrity of a digital object but does not track what changed.

According to the State Archives the advantages of using SIMPLE significantly surpass the drawbacks. The following the advantages were cited.

1. Minimal Resources Needed: Even a small archives can operate SIMPLE without the need to allocate significant additional resources (monetary and staff).

2. Embedding of Arbitrary Metadata: Metadata can be added to any digital object to show that it was generated by the State Archives and to which holding it belongs.

3. Clear Text Metadata Bound with the Digital Object: The metadata will travel with the digital object and can be read by anyone irrespective of the format of the digital object.

4. Verifiability of Integrity: The integrity can be verified at any point in time.

[2]See OAIS reference model [CCSDS, 2002].

5. Ease of Integration into an Automated Process: The step of adding SIMPLE metadata to a digital object was easily added to the migration process.
6. Traceability of the Migration Process: Every step of the migration process is documented by intermediary results and log files that allow the traceability of every step of the migration.
7. More Know-How and Experience in Digital Archiving: Thanks to the debates on the subject on SIMPLE and its deployment, the staff of the State Archives has gained a better understanding of digital archiving. This has led to a constant amelioration in the operation of the digital archives.
8. Universal Usability of SIMPLE: All holdings would benefit from the use of SIMPLE, namely by providing identification for digital objects, securing their integrity and making changes traceable through lineage.
9. Reversibility: The process of adding SIMPLE metadata to a digital object can be reversed if needed.
10. Small Storage Overhead: The metadata generated by SIMPLE is negligible in comparison to the data volume. The storage requirements will only grow moderately when employing SIMPLE.
11. No objections by the IT department: The IT department had no objections that the State Archives operate the SIMPLE prototype in the cantonal IT environment.

The State Archives declare that all their goals have been achieved. However, this cannot yet be stated with confidence, since, while the files have been disseminated, no requests based on these files have been received to date. Therefore, the question whether or not the SIMPLE metadata will be available when a request is made to the archives cannot be answered at present.

5.5 Summary

In this chapter, we have shown an implementation of the SIMPLE method: the SIMPLE prototype. We have presented the functionality, interfaces, and implementation of the prototype and compared it to the requirements as defined by the SIMPLE method. The practical use and usability of the prototype have been shown in a case study at the State Archives of Appenzell Ausserrhoden. In this case study, the SIMPLE method and prototype allow the traceable migration of scanned family records and will support users in locating the correct context for distributed digital objects in the future.

In the next chapter, we will draw conclusions from the work presented in this thesis, stressing its contributions and highlighting its limitations and the possibilities for future work on what has been presented here.

6

Limitations and Future Work

The SIMPLE method focuses on digital objects on the bit level. This focus reduces the possibilities to introduce additional format specific characteristics into the equality and integrity considerations. Even the smallest change to the digital object will lead to a different digital object which is not equal to the unchanged object. Even so, these small changes might not be of importance for the digital object format. We believe the format specific normalization of digital objects to be an interesting topic for future research.

Focus on the digital object means that the relations to other digital objects referenced by the digital object are ignored. Since digital objects often reference other digital objects that are later used to render the referenced digital objects, the referenced objects form an important part of the rendered digital object. Our approach currently only includes ancestors of digital objects (and their identification), but not references to other digital objects. However, since any referenced digital object may change, this would significantly alter the rendered digital object if the rendering process were using the referenced digital objects as input. Future work will include research on the detection of references in digital objects as well as their inclusion in the SIMPLE metadata. It should be noted that this inclusion will be recursive, as the referenced digital objects will again reference digital objects.

Ancestry for digital objects only supports digital objects as ancestors. All other inputs for digital object creation are ignored and will not be represented in the ancestry graph. However, many digital objects stem from physical objects, an ancestry relation which cannot be represented with our current model. A scanned photo is an example of an ancestry relation between a physical photo print and a digital object representing it.

The identifiers used by the SIMPLE method are embedded in existing identifier systems, thereby benefiting from the infrastructure that is already in place. However, the use of existing identifier systems limits the use of the identifier functions used in the SIMPLE method, the cryptographic hash functions. Additionally, due to space requirements, the existing identifier system may not accommodate the multiple identifiers which SIMPLE supports. We therefore believe that identifier systems that support multiple identifiers are an interesting topic for future research.

In the field of metadata formats, the SIMPLE method gives the applicant a wide range of options on how to represent the SIMPLE metadata within the SIM XML container. Since there are

no defined formats for the metadata parts of lineage, integrity, traceability, and documentation the mapping of the metadata to XML will vary. Additionally, the processing of metadata was beyond the scope of this thesis. Research on the formal definition of the processing history of a digital object is of vital interest, not only within the scope of SIMPLE, but for any application with traceability requirements.

Sealing digital objects while providing a useful tool to detect deviations from integrity may lead to a false sense of security. Digital objects can be sealed at any time, giving no reliable indication of the time or entity performing the sealing.

The SIMPLE method defines some rigid outlines which allow a straightforward implementation. Some of these outlines may be too rigid and they may hinder the adoption of the SIMPLE method. We believe that the following issues need closer inspection for the specific reasons given in each situation.

- Complete lineage capture: Frequent changes to the metadata within the digital object lead to a fine grained lineage graph and to many intermediate digital objects which only offer a small contribution to the evolution of the digital object. Additionally, privacy and trade secret concerns stemming from the retracing of all processing, as well as the storage overhead associated with the complete capture of the lineage, show the need for incomplete or partial lineage.

- Encoding of SIMPLE metadata: The SIMPLE metadata is encoded in UTF-8 to provide an easily recognizable byte sequence for the identification of the metadata start and end. This limitation may lead to problems in the format of the digital object, for example if the UTF-8 encoded metadata is embedded in an UTF-16 encoded XML document.

- Finite and complete digital objects: In its current form, the SIMPLE method only supports finite and complete digital objects. However, identification and integrity, as well as ancestry might be useful for parts of digital objects as well as digital objects which are not yet complete, for example a video broadcast.

- Dependence on digital object formats: The integration of SIMPLE metadata with the digital object depends on the availability of unrestricted or undefined spaces within the format definition of the digital object formats. This leads to a fragile link between the metadata and the digital object, since the format will not recognize the SIMPLE metadata and can therefore generally not provide access to it.

- Identification is not part of the digital object: The identification of a digital object, as defined in the SIMPLE method, cannot be stored within the digital object itself, since this alters the identification. Therefore, the identifier of a digital object can only be derived if the entire digital object is accessible. However, if the digital object is sealed, the identifier of the unsealed digital object is stored within, thereby providing the sought identifier.

- Ancestry graphs are restricted to lineage: The ancestry graph of a digital object is restricted to a lineage graph in the SIMPLE method. Therefore, the ancestry graph will only represent an extract of the complete ancestry graph. Additionally, with the SIMPLE prototype, the ancestry graph was not mapped to XML.
- Sealing of digital objects: The sealing process used is specific to the SIMPLE method and does not use digital signatures. Other approaches to ensure the integrity of the digital object should be taken into consideration.

The SIMPLE prototype has shown its usefulness as well as that of the SIMPLE method in general in the case study using the State Archives of Appenzell Ausserrhoden. For a more widespread usage, the following features would have to be ameliorated. Currently, only digital objects in the form of files in a file system can be the starting point for the prototype. Digital objects are, however, defined in a much broader context. In the future, arbitrary digital objects should be processed by a SIMPLE application, including, for example, remote resources like web pages or binary objects in a database. The user interface, specifically the web and the bulk processing interface, needs to be extended to include all functionality provided by the SIMPLE library. Furthermore, this library needs a clearer structure and defined APIs (application programming interfaces). In that way, it can be made available for incorporation in other applications which can then use the functionality provided in a clearly defined manner. Currently, the format of the digital object is determined from the file name suffix of the file name of the digital object. In view of the ambiguity of such a mapping, as well as the disentanglement of the digital object from the file system, the format of the digital object should, in the future, be determined from the contents of the digital object thus emphasizing the independence of the digital object from its environment. However, the main enhancements sought by the users of the SIMPLE prototype are a better integration of the SIMPLE metadata with the digital object so that it will survive the most common processing, as well as easier access to the metadata for the recipients of the digital objects containing SIMPLE metadata.

We believe that the SIMPLE method provides a useful starting point for future research on digital object autonomy, integrity, referenceability, and ancestry.

7

Conclusions

Which digital objects were used to create the digital object at hand? The methods and tools developed in this thesis aid in answering this question. Based on the formal definition of the identification of and equality between digital objects, the ancestry of a digital object can be recorded unambiguously.

The first step in answering the question above is the unambiguous identification of the ancestors of the digital object. We define the identity of the digital object by its contents, that is, by the series of bits that make up a digital object. Equality is defined, likewise, to the point that every bit of two digital objects must be equal for them to be equal. Thus, the identity of the digital object is no longer defined by access paths like URLs or file system paths. By freeing a digital object's identity from the information system managing it, the life of a digital object is no longer tied to that of one information system, it may continue to exist in other information systems while retaining its identity, thereby enabling access longer than the life of any information system. The unambiguous identification of a digital object is the prerequisite for any persistent reference. The identity of a digital object is established by applying an identification function to the contents of the digital object. We used cryptographic hash functions on the digital object to establish their identity, for they are well researched and established, and are also suited to assess the integrity of digital objects due to their collision resistance.

The second step in answering the question above is the determination of the ancestors of a digital object. The ancestors can only be determined if their parenthood can be established. To date, the ancestry of the digital object cannot be established nor can it be recorded, except in specialized version control or workflow systems. To resolve this, we described a method called *SIMPLE* (*S*imple *I*dentifiable *M*etadata with *P*ersistent *L*ineage *E*mbedding), which enables the unobtrusive addition of arbitrary metadata to digital objects and thereby the inclusion of ancestry metadata.

Establishing the ancestry of a digital object is the overarching aim of the SIMPLE method. This aim is broken down into the following requirements:

- Information system independence, so the ancestry does not need be determined by a specific information system

- Metadata to digital object adhesion, so the ancestry cannot be separated from the digital object easily
- Metadata identification and access, so the ancestry metadata can be identified and accessed independent of the type of digital object
- Integrity, so the ancestry cannot be changed inadvertently
- Explicit ancestry relationships, so the ancestry is recorded; arbitrary metadata inclusion to enable extensibility of the metadata
- Software and technology invariance, so the recorded ancestry will not be lost in the next software or technology update
- Minimal complexity, so the metadata is easy to understand

The requirements have been implemented by using a defined, self-documenting XML container, which is merged with the digital object. The XML container can be located inside any digital object, irrespective of the information system. Depending on the format of the digital object the XML container can be embedded in a way that allows processing of the digital object as if the XML container were not present; for other formats the XML container may have to be removed to enable further processing. Within the XML container, arbitrary metadata can be inserted. We inserted ancestry and integrity metadata, thereby covering the ancestry and integrity requirements. The integrity of the digital object is established through a "sealing" process. However, this does not guard against malicious attacks against the integrity. Attacks of this kind cannot be detected without the use of information external to the digital object. We propose to use the identifier as external integrity information, since the identifier will change whenever the digital object changes. Our method is not technology invariant, since we employ XML as technology. However, the XML metadata can be converted to any format by using XSL transformation, thereby providing an exit strategy for this technology dependence.

A case study with the State Archives of Appenzell Ausserrhoden has shown that the SIMPLE method and its implementation provided added value for a small scale migration of a digital archives' fonds. The continued feedback from the State Archives has been an important source for the definition of the requirements. The results from the case study are very encouraging. For the future, the main wishes for enhancement by the users in our study point towards an integration of existing information systems as a source for digital object metadata, as well as easier access to the metadata.

We believe that this work is just the beginning of the discussion of how digital objects are processed and used to create new digital objects with lineage. With the continuing rise in importance of copyright and the citation of non-article sources this question is likely to gain more and more weight.

List of Figures

List of Tables

Bibliography

[21C, 1997] (1997). 21 cfr part 11. In *Federal Register*, number 62 in 54, pages 13429–13466. http://www.fda.gov/ora/compliance_ref/part11/FRs/background/pt11finr.pdf, last retrieved 2007-03-10.

[Adobe, 2005] Adobe (2005). Xmp specification. Specification, Adobe Systems Incorporated. http://www.adobe.com/devnet/xmp/pdfs/xmp_specification.pdf, last retrieved 2008-06-18.

[Allen et al., 2007] Allen, J. D., Becker, J., Cook, R., Davis, M., Everson, M., Freytag, A., Jenkins, J. H., Ksar, M., McGowan, R., Moore, L., Muller, E., Scherer, M., Suignard, M., and Whistler, K., editors (2007). *The Unicode Standard, Version 5.0*. Addison-Wesley, Boston, MA, USA. Electronic edition http://www.unicode.org/versions/Unicode5.0.0/, last retrieved 2008-12-13.

[Altman and King, 2007] Altman, M. and King, G. (2007). A proposed standard for the scholarly citation of quantitative data. *D-Lib Magazine*, 13(3/4). http://www.dlib.org/dlib/march07/altman/03altman.html, last retrieved 2008-04-22.

[Apache Ant, 2008] Apache Ant (2008). Website. http://ant.apache.org, last retrieved 2008-12-13.

[Appenzell Ausserrhoden, 2008] Appenzell Ausserrhoden (2008). Staatsarchiv - appenzell ausserrhoden. Website. http://www.ar.ch/departemente/kantonskanzlei/staatsarchiv/, last retrieved 2008-12-13.

[Arms et al., 1997] Arms, W. Y., Blanchi, C., and Overly, E. A. (1997). An architecture for information in digital libraries. *D-Lib Magazine*. http://www.dlib.org/dlib/february97/cnri/02arms1.html, last retrieved 2007-11-10.

[Bara, 1995] Bara, B. G. (1995). *Cognitive Science: A Developmental Approach To The Simulation Of The Mind*. Psychology Press (UK).

[Bennett, 1997] Bennett, J. C. (1997). A framework of data types and formats, and issues affecting the long term preservation of digital material. Report 50, British Library Research and Innovation Centre. http://www.ukoln.ac.uk/services/elib/papers/supporting/pdf/rept011.pdf, last retrieved 2008-06-16.

[Berners-Lee et al., 2005] Berners-Lee, T., Fielding, R. T., and Masinter, L. (2005). Uniform resource identifier (uri): Generic syntax. Request for Comments 3986, The Internet Society. http://www.ietf.org/rfc/rfc3986.txt, last retrieved 2007-02-28.

[Bos et al., 2007] Bos, B., Çelik, T., Hickson, I., and Lie, H. W. (2007). Cascading style sheets level 2 revision 1 (css 2.1) specification. Website. http://www.w3.org/TR/CSS21/, last retrieved 2008-12-14.

[Bose, 2004] Bose, R. (2004). *Composing and Conveying Lineage Metadata for Environmental Science Research Computing*. PhD thesis, Bren School of Environmental Science and Management, University of California, Santa Barbara. http://homepages.inf.ed.ac.uk/rbose/pubs/rkb_dissertation_final_protected.pdf, last retrieved 2007-03-10.

[Boudrez, 2005] Boudrez, F. (2005). Digitale containers voor het digitaal archiefdepot. Digitaal depot, Expertisecentrum DAVID. http://www.expertisecentrumdavid.be/docs/digitale_containers.pdf, last accessed: 2007-03-08.

[Bray et al., 2004] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., and Cowan, J. (2004). *Extensible Markup Language (XML) 1.1*. World Wide Web Consortium (W3C). http://www.w3.org/TR/xml11/, last retrieved 2006-01-09.

[Buneman et al., 2000] Buneman, P., Khanna, S., and Tan, W. C. (2000). Data provenance: Some basic issues. In Kapoor, S. and Prasad, S., editors, *FSTTCS*, volume 1974 of *Lecture Notes in Computer Science*, pages 87–93. Springer.

[Buneman et al., 2001] Buneman, P., Khanna, S., and Tan, W. C. (2001). Why and where: A characterization of data provenance. In den Bussche, J. V. and Vianu, V., editors, *ICDT*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer.

[CASPAR project, 2009] CASPAR project (2009). The caspar project - cultural, artistic and scientific knowledge for preservation, access and retrieval. Website. http://www.casparpreserves.eu/caspar-project, last retrieved 2009-10-22.

[CCSDS, 2002] CCSDS (2002). Consultative committee for space data systems: Reference model for an open archival information system (oais), blue book, issue 1, january 2002. http://public.ccsds.org/publications/archive/650x0b1.pdf, last retrieved 2004-10-08, [Equivalent to ISO 14721:2003].

[CCSDS, 2007] CCSDS (2007). Consultative committee for space data systems: Xml formatted data unit (xfdu) structure and construction rules, red book, january

2007. http://public.ccsds.org/sites/cwe/rids/Lists/CCSDS%206610R1/Attachments/661x0r1.pdf, last retrieved 2007-09-21.

[CCSDS, 2008] CCSDS (2008). Consultative committee for space data systems: Xml formatted data unit (xfdu) structure and construction rules, blue book, issue 1, september 2008. http://public.ccsds.org/publications/archive/661x0b1.pdf, last retrieved 2008-12-13.

[CitesSeer, 2008] CitesSeer (2008). Website. http://citeseer.ist.psu.edu/citeseer.html, last retrieved 2008-12-13.

[Crespo and Garcia-Molina, 1998] Crespo, A. and Garcia-Molina, H. (1998). Archival storage for digital libraries. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 69–78, New York, NY, USA. ACM.

[Cundiff, 2004] Cundiff, M. V. (2004). An introduction to the metadata encoding and transmission standard (mets). *Library Hi Tech*, 22(1):52–64. http://www.ietf.org/rfc/rfc3275.txt, last retrieved 2008-12-06.

[Diestel, 2005] Diestel, R. (2005). *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg.

[DSpace, 2008] DSpace (2008). Website. http://www.dspace.org/, last retrieved 2008-12-13.

[Duranti, 1998] Duranti, L. (1998). *Diplomatics: New Uses for An Old Science*. Society of American Archivists and Association of Canadian Archivists in association with Scarecrow Press.

[Duval et al., 2002] Duval, E., Hodgins, W., Sutton, S. A., and Weibel, S. (2002). Metadata principles and practicalities. *D-Lib Magazine*, 8(4). http://www.dlib.org/dlib/april02/weibel/04weibel.html, last retrieved 2007-03-10.

[e-Depot, 2008] e-Depot (2008). Website. http://www.kb.nl/dnp/e-depot/e-depot-en.html, last retrieved 2008-12-13.

[Eastlake et al., 2002] Eastlake, D., Reagle, J., and Solo, D. (2002). Xml-signature syntax and processing. Request for Comments 3275, The Internet Society. http://www.ietf.org/rfc/rfc3275.txt, last retrieved 2007-02-28.

[EMC Corporation, 2007] EMC Corporation (2007). Emc centera content-addressed storage system. Data Sheet C931.7, EMC Corporation. http://www.emc.com/collateral/hardware/data-sheetcentera-ds.pdf, last retrieved 2008-03-10.

[Francis et al., 1998] Francis, R., Gibbs, R., Harari, L., Heazlewood, J., Hills, B., Leask, N., Sefton, A., Waugh, A., and Wilkinson, R. (1998). Electronic archiving - a 100 year experiment. In *Proceedings of the Third Document Computing Symposium*, Sydney, Australia. http://www.prov.vic.gov.au/vers/publications/pdf/Electronic%20Archiving.pdf, last retrieved 2007-03-10.

[GIT, 2009] GIT (2009). git - the fast version control system. Website. http://git-scm.com/, last retrieved 2009-10-22.

[Gladney, 2004] Gladney, H. M. (2004). Trustworthy 100-year digital objects: Evidence after every witness is dead. *ACM Trans. Inf. Syst.*, 22(3):406–436.

[Gladney, 2007] Gladney, H. M. (2007). *Preserving Digital Information*. Springer.

[Grant, 2009] Grant, R. (2009). Filesystem interface for the git version control system - final report. http://www.seas.upenn.edu/~cse400/CSE400_2008_2009/websites/grant/final.pdf, last retrieved 2009-10-22.

[Hedstrom and Lee, 2002] Hedstrom, M. and Lee, C. A. (2002). Significant properties of digital objects: Definitions, applications, implications. In *Proceedings of the DLM-Forum*, pages 218–223. http://www.ils.unc.edu/callee/sigprops_dlm2002.pdf, last retrieved 2008-03-02.

[Heslop et al., 2002] Heslop, H., Davis, S., and Wilson, A. (2002). An approach to the preservation of digital records. http://www.naa.gov.au/recordkeeping/er/digital_preservation/Green_Paper.pdf, last retrieved 2007-03-10.

[Heuscher, 2003] Heuscher, S. (2003). Softening the borderlines of archives through xml - a case study. *Archivi & Computer*, 1-2:116–122.

[Heuscher, 2006] Heuscher, S. (2006). Workflow modeling language evaluation for an archival environment. In *Archivi Informatici per il Patrimonio Cultutale, 17-19 November 2003, Accademia Nationale Dei Lincei, Rome, Italy*, number 114, pages 131–153.

[Heuscher et al., 2004] Heuscher, S., Jaermann, S., Keller-Marxer, P., and Moehle, F. (2004). Providing authentic long-term archival access to complex relational data. In *Proceedings PV-2004: Ensuring the Long-Term Preservation and Adding Value to the Scientific and Technical Data, 5-7 October 2004, ESA/ESRIN, Frascati, Italy, Noordwijk: European Space Agency*, pages 241–261.

[International Council on Archives, 1999] International Council on Archives (1999). Isad(g): General international standard archival description, 2nd edn. ICA Standards. http://www.ica.org/biblio/isad_g_2e.pdf, last retrieved 2005-02-10.

[ISO, 2005] ISO (2005). International organization for standardization: Document management – electronic document file format for long-term preservation – part 1: Use of pdf 1.4 (pdf/a-1). Standard 19005-1:2005, ISO, Geneva, Switzerland.

[JHOVE, 2008] JHOVE (2008). Jstor/harvard object validation environment. Website. http://hul.harvard.edu/jhove/, last retrieved 2008-12-13.

[Josefsson, 2006] Josefsson, S. (2006). The base16, base32, and base64 data encodings. Request for Comments 4648, The Internet Society. http://tools.ietf.org/rfc/rfc4648.txt, last retrieved 2007-01-26.

[Kindberg, 2002] Kindberg, T. (2002). Implementing physical hyperlinks using ubiquitous identifier resolution. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 191–199, New York, NY, USA. ACM. http://doi.acm.org/10.1145/511446.511472, last retrieved 2008-02-15.

[Kunze and Rodgers, 2007] Kunze, J. and Rodgers, R. P. C. (2007). The ark persistent identifier scheme. Internet-draft, Internet Engineering Task Force. http://www.ietf.org/internet-drafts/draft-kunze-ark-14.txt, last retrieved 2008-02-15.

[LOC, 2002] LOC (2002). The library of congress: Encoded archival description (ead). Website. http://www.loc.gov/ead/, last retrieved 2008-12-14.

[LOC, 2005] LOC (2005). The library of congress: Metadata encoding and transmission standard (mets). http://www.loc.gov/standards/mets/, last retrieved 2005-12-19.

[Lorie, 2001] Lorie, R. A. (2001). Long term preservation of digital information. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 346–352, New York, NY, USA. ACM.

[McCarron and Ishikawa, 2007] McCarron, S. and Ishikawa, M. (2007). Xhtml 1.1 - module-based xhtml - second edition. Website. http://www.w3.org/TR/xhtml11/, last retrieved 2008-12-14.

[Menezes et al., 1997] Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1997). *Handbook of Applied Cryptography*. CRC Press, 5 edition. http://www.cacr.math.uwaterloo.ca/hac/, last retrieved 2008-04-20.

[METS Editorial Board, 2006] METS Editorial Board (2006). *<METS> - Metadata Encoding and Transmition Standard: Primer and Reference Manual*. Digital Library Federation, draft edition. http://www.loc.gov/standards/mets/METSDocumentationdraft8.30.pdf, last retrieved 2007-02-28.

[Mironov, 2005] Mironov, I. (2005). Hash functions: Theory, attacks, and applications. Technical Report MSR-TR-2005-187, Microsoft Research. ftp://ftp.research.microsoft.com/pub/tr/TR-2005-187.pdf, last retrieved 2008-03-09.

[Naor and Yung, 1989] Naor, M. and Yung, M. (1989). Universal one-way hash functions and their cryptographic applications. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43, New York, NY, USA. ACM.

[Paskin, 2007] Paskin, N. (2007). Digital object identifier (doi™) system. *Encyclopedia of Library and Information Sciences*, (3). http://www.doi.org/overview/070710-Overview.pdf, last retrieved 2008-12-06, forthcoming.

[Pelegrí-Llopart et al., 1999] Pelegrí-Llopart, E., Cable, L., and Ahmed, S. (1999). Javaserver pages specification. Website. `http://java.sun.com/products/jsp/archive.html`, last retrieved 2008-12-14.

[PREMIS Working Group, 2005] PREMIS Working Group (2005). Data dictionary for preservation metadata: Final report of the premis working group. Report, OCLC and RLG. `http://www.oclc.org/research/projects/pmwg/premis-final.pdf`, last retrieved 2007-11-10.

[Preneel, 1993] Preneel, B. (1993). *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven. `https://www.cosic.esat.kuleuven.be/publications/thesis-2.pdf`, last retrieved 2008-03-10.

[Quenault, 2004] Quenault, H. (2004). Vers: Building a digital record heritage. In *Proceedings of IS&T 2004 Archiving Conference, San Antonio, Texas, USA, 20. April 2004*, volume 1, pages 2–7.

[Quinlan and Dorward, 2002] Quinlan, S. and Dorward, S. (2002). Venti: A new approach to archival data storage. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 7, Berkeley, CA, USA. USENIX Association.

[Raymond, 2003] Raymond, E. S. (2003). *The Art of UNIX Programming*. Addison-Wesley Professional Computing Series. Addison Wesley Professional, 1 edition.

[Schneider et al., 2005] Schneider, J., Yu, R., and Dyer, J. (2005). Ecmascript for xml (e4x) specification. Website. `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-357.pdf`, last retrieved 2008-12-14.

[Shafer et al., 1996] Shafer, K., Weibel, S., Jul, E., and Fausey, J. (1996). Introduction to persistent uniform resource locators. `http://purl.oclc.org/docs/inet96.html`, last retrieved 2008-02-22.

[Spéry et al., 2001] Spéry, L., Claramunt, C., and Libourel, T. (2001). A spatio-temporal model for the manipulation of lineage metadata. *GeoInformatica*, 5:51–70.

[Sun et al., 2003] Sun, S., Lannom, L., and Boesch, B. (2003). Handle system overview. Request for Comments 3650, The Internet Society. `http://www.ietf.org/rfc/rfc3650.txt`, last retrieved 2008-02-15.

[Sun Microsystems Incorporated, 2008] Sun Microsystems Incorporated (2008). Website. `http://java.sun.com/j2se/1.4.2/`, last retrieved 2008-12-14.

[Vermeulen et al., 1999] Vermeulen, A., Ambler, S. W., Bumgardner, G., Metz, E., Misfeldt, T., and Shur, J. (1999). *The Elements of Java Style*. Cambridge University Press.

[Waugh et al., 2000] Waugh, A., Wilkinson, R., Hills, B., and Dell'oro, J. (2000). Preserving digital information forever. pages 175–184.

[Widom, 2005] Widom, J. (2005). Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276.

[Williamson et al., 2002] Williamson, A., Wu, A., Gibson, J., and Pepperdine, K. (2002). *Ant Developer's Handbook*. Sams Publishing.

[Wilson, 2007] Wilson, A. (2007). Significant properties report. Inspect work package 2.2, Arts and Humanities Data Service, London, UK. `http://www.significantproperties.org.uk/documents/wp22_significant_properties.pdf`, last retrieved 2008-12-13.

[Woodruff and Stonebraker, 1997] Woodruff, A. and Stonebraker, M. (1997). Supporting fine-grained data lineage in a database visualization environment. In Gray, W. A. and Larson, P.-Å., editors, *ICDE*, pages 91–102. IEEE Computer Society.

[Yergeau, 1998] Yergeau, F. (1998). Utf-8, a transformation format of iso 10646. Request for Comments 2279, The Internet Society. `http://www.ietf.org/rfc/rfc2279.txt`, last retrieved 2008-06-17.

[You et al., 2005] You, L. L., Pollack, K. T., and Long, D. D. E. (2005). Deep store: An archival storage system architecture. *icde*, 0:804–8015.

A

Methodology for the Evaluation of the SIMPLE Case Study

The State archivist and the archivist of Appenzell Ausserrhoden answered the following questions in order to allow the evaluation of the case study. The State archivist is in charge of the State Archives, while the archivist put the case study into practice. The respondent for each question is marked in brackets. Both interviews were consecutively held at the State Archives in Herisau on the 5. November 2009.

1. Please tell us about your career, how you came to the State Archives of Appenzell Ausserrhoden, and digital archiving. (State archivist and archivist)
2. What is the mission of the State Archives of Appenzell Ausserrhoden? (State archivist)
3. What is the size of your archives, paper and digital? (State archivist)
4. How many people are employed at the State Archives and what is the budget of the archives? (State archivist)
5. How was the digital archives started? (State archivist and archivist)
6. What are the challenges specific to digital holdings, compared to traditional holdings? (State archivist and archivist)
7. How does the preservation of digital holdings compare to traditional holdings? How do you asses the condition of a digital object? (State archivist and archivist)
8. What areas do you see your need for additional support and/or training in the digital domain? (State archivist and archivist)
9. What was your first contact with SIMPLE and what was your first impression? (State archivist and archivist)
10. How does SIMPLE work? (State archivist and archivist)
11. Why was SIMPLE used in the project? (State archivist and archivist)

12. What digital collections will next be migrated using SIMPLE? Which collections are not suited for SIMPLE? (State archivist and archivist)
13. What are the arguments against and for SIMPLE? (State archivist and archivist)
14. What is missing in SIMPLE? What suggestions for improvements do you have? (State archivist and archivist)
15. How was SIMPLE used in the migration? (Archivist)
16. Have you got any concluding remarks? (State archivist and archivist)

Questions 1 to 4 concern personal matters and matters concerning the State Archives. Questions 5 to 8 concern the digital archives within the State Archives. Finally, questions 9 to 16 concern the SIMPLE concept and prototype.

Curriculum Vitae

Last Name	Heuscher
First Names	*Stephan* Jakob Benedikt
Year of Birth	1974
Place of Origin	Herisau, Appenzell Ausserrhoden, Switzerland
1980 – 1986	Primary schools in Umiken and Rombach, Switzerland and Scarsdale, USA
1986 – 1994	Gymnasii in Aarau and Bern, Switzerland
1994 – 1999	Master studies at the Department of Electrical Engineering at the ETH in Zürich, Switzerland
1999 – 2000	Research and Development at Swisscom AG in Bern, Switzerland
2000 – 2001	Crosspoint Informatik AG in Schönbühl, Switzerland
2002 – 2004	Swiss Federal Archives in Bern, Switzerland
2002 – 2010	PhD studies at the Department of Informatics at the University of Zurich, Switzerland
2004 – 2005	ikeep AG in Bern, Switzerland
2005 – 2006	REAG IT-Solutions AG in Muri bei Bern, Switzerland
2006 –	Bedag Informatik AG in Bern, Switzerland